EFFICIENT DATA TO DECISION PIPELINES FOR EMBEDDED AND SOCIAL SENSING

BY

HIEU KHAC LE

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2012

Urbana, Illinois

Doctoral Committee:

Professor Tarek F. Abdelzaher, Chair
Professor Klara Nahrstedt
Professor Dan Roth
Professor Boleslaw Szymanski, Rensselaer Polytechnic Institute

ABSTRACT

This dissertation presents results of our studies in making data to decision pipelines for embedded and social sensing efficient. Due to the pervasive presence of wired sensors, wireless sensors, and mobible devices, the amount of data about the physical world (environmental measurements, traffic, etc.) and human societies (news, trends, conversations, intelligence reports, etc.) reaches an unprecedented rate and volume. This motivates us to optimize the way information is collected from sensors and social entities. Two challenges are addressed: (i) How can we gather data such that throughput is maximized given the physical constraints of the communication medium? and (ii) How can we process inherently unreliable data, generated by large networks of information and social sources? We present some essential solutions addressing these challenges in this dissertation. The dissertation is organized in two parts. Part I presents our solution to maximizing bit-level data throughput by utilizing multiple radio channels in applications equiped with wireless sensors. Part II presents our solution to dealing with the large amount of information contributed by unvetted sources.

To mom and dad.

ACKNOWLEDGMENTS

First, I would like to thank my compassionate advisor, Professor Tarek F. Abdelzaher for his constant support and encouragement. I feel extremely graceful to have him as my advisor. He not only teach me how to navigate through my academic life; but he also quitely, and perhaps unconciously, make me a nicer person along the way. I own you a good use of my degree for your kindness and faith.

Thanks to my parents for their consistent and gentle reminder to complete my study. I also thanks them for being patient and understanding for my rare visits during my study time. I would like to thanks my brothers and my sisters-in-law to take care of my parents during all these years. I love you all.

Thanks to my friends (Ly Le, Nhung Nguyen, Thien Nguyen, Dung Nguyen, Kim Pham, Kailin Liu, and others) who make the lonely process of pursusing a PhD more managable. After this long journey, I understand better how valuable friendships are for ones wanting to tread a long path. Thanks for being with me.

Thanks to Caterva team, Caterva investors, Mr. Rob Schultz, and Mr. Bill Tai for giving me hope and joy working with them during my PhD years. I own you guys a 110% focus on the company.

Thanks to Professor Klara Nahrstedt, Professor Dan Roth, and Professor Boleslaw Szymanski for serving on my PhD commitee, for your valuable time and feedback to make the thesis more focused, concise; which helps to highlight the main contribution of my PhD works better.

Thanks to Vietnam Education Foundation for the support during my graduate study. I would like to refer to you as a collection of people that I was fortunate to interact with: Professor John Hopcroft, Professor Alfred Aho, Mr. Kien Pham, Dr. Lynn McNamara, Ms. Sandarshi Gunawardena, and so many other individuals that help me along the way. Thank you.

Thanks to following collaborators who make the research in this dissertation possible: Professor Tarek Abdelzaher, Dan Henricksson, Dong Wang, Hossein Ahmadi, Md Yusuf S Uddin, Md Tanvir Al Amin, Omid Fatemieh, Professor Dan Roth, Professor Boleslaw Szymanski, Tommy Nguyen, Professor Sibel Adali, Charu Aggarwal, Jin Heo, Raghu Ganti, Professor Jiawei Han, Dr. Lance Kaplan, Yo Han Ko, and Jeff Pasternack.

Finally, thank God for making this part of my life experience. During these years, I have learned and grown a lot.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Current work on sensor networks and general networking literature focuses on maximizing classical network-level performance metrics such as delay or throughput. In contrast, this thesis takes an end-to-end approach that improves efficiency of *information delivery* to the recepient at different stages of a "data to decision" pipeline. Efficient information delivery in such a pipeline implies more than just throughput maximization but also includes maximizing quality of information delivered for decsion-making We further generalize work on sensing to include collection of information from social sources (in addition to physical sensors).

Different from the traditional model of sensing, in social and information centric sensing, information and elements of social networks are considered as first class abstractions. The objective of designing such a system is to maximize the net actionable information delivered at the very end of the processing chain to a decision-maker. Guided by this goal, common services which operate directly on information and social network abstractions are needed. These services are important since they allow system builders to design more sophisticated applications with less concern about low-level mechanisms for data collection, cleaning, processing, and resource optimization. Thus, the application builders can focus on high level information manipulation instead of optimizing low-level performance. Social and information centric sensing push the frontier of sensing towards more integrated and jointly optimized approaches across three important areas: communication networks, where data throughput is the main focus; information networks, where actionable information is the main focus; and social networks, where social sources and their relationships are the main

focus. In this thesis, we consider optimizations at two different stages of the data to decision pipeline:

- At the first stage, where data is collected from the environment (e.g., via sensors, or via mobile devices operated by humans), we consider optimizing bit throughput. This part is the more conventional and was done first. It is concerned with protocols that optimize data collection systems.

- In the second stage, sensing applications handle information successfully arriving at the backend. This stage is where algorithms are employed to extract high-quality information from the myriads of collected data items generated by vast number of users or sensors. A challenge at this stage is to know which pieces of information are more important, to invest more computing power on (by applying more expensive algorithms), and which are less important that one can simply ignore.

The goal of this dissertation is to develop efficient techniques for collecting, handling, and processing large amounts of information from sensors and social network sources. Specifically, the main contributions of this dissertation are two-fold. First, we investigate algorithms and protocols to make the communication layer able to collect more data given radio constraints. Second, we propose a new service to convert large amounts of low-quality data into a manageable amount of higher-quality information.

The dissertation is organized as follows. In Part I, we present our research results in maximizing communication throughput in networks of standard wireless sensing devices equiped with a single half-duplex radio. In Part II, we present Apollo, a data distillation service which allows the backend to efficiently select pieces of information that are more valuable based on joint credibility assessment of information and its sources.

PART I

Communication Layer

A "data-to-decision" information pipeline starts with collecting the raw data. In this section, we address the efficiency of data collection using networked sensors. Devices equipped with sensors and wireless radios have an increasingly pervasive presence. However, due to the inherent physical characteristics of the wireless communication medium, there is a limit on how much data can be tranfered among these devices within a given space and time. Additionaly, due to energy limitations, these devices usually operate using simple and energy-minimal hardware. For example, they most likely have only one radio device which can operate on one frequency at a time. In this part, we present our approaches for improving data collection throughput, and develop protocols to efficiently utilize multiple radio channels simultaneously.

This Part is organized as follows. In Chapter 2, we present a solution to increase thoughput by utilizing multi-channel communication for the most common and simple type of wireless sensing applications: data collection. In Chapter 3, we present a more generic solution for sensing applications using the same hardware configuration but with an arbitrary traffic pattern.

CHAPTER 2

A CONTROL THEORY APPROACH TO THROUGHPUT OPTIMIZATION IN MULTI-CHANNEL COLLECTION SENSOR NETWORKS

Most currently deployed sensor networks use the same channel to communicate information among nodes. This is a source of great inefficiency as it poorly utilizes the available wireless spectrum. This study takes advantage of radio capabilities of MicaZ motes that can communicate on multiple frequencies as specified in the 802.15.4 standard. We consider the case of a data collection sensor network where multiple base-stations are responsible for draining data from sensor nodes. A key question becomes how to assign nodes to wireless channels such that network throughput is maximized. The problem is reduced to one of load balancing. A control theoretical approach is used to design a self-regulating load-balancing algorithm that maximizes total network throughput. It is evaluated both in simulation and on an experimental testbed. The results demonstrate a significant performance improvement. It is shown that a control theory approach is indeed needed to guarantee stability in data collection networks and prevent undue oscillation of nodes among different wireless channels upon dynamic changes in load conditions.

2.1 Introduction

This chapter introduces a control-theoretic approach for maximizing throughput in multi-channel sensor networks. The main problem addressed is one of designing distributed algorithms for choosing node communication frequencies such that the total network throughput is maximized. It is shown that delay incurred in propagating network state can cause instability manifested in frequent switching of nodes among different frequencies, which reduces network throughput.
The chapter derives stability conditions and shows that the resulting algorithm substantially improves throughput over the selected baselines.

A significant number of present sensor node prototypes use radio modules capable of transmission on multiple channels, such as the 802.15.4 radio. While a plethora of MAC-layer protocols have been designed for sensor networks, implemented protocols have all featured a single channel assumption. Throughput-maximizing dynamic assignment of nodes to channels has not been addressed.

An argument in favor of existing MAC-layer protocols is that most current sensor networks carry only very limited traffic, such as single integer values of infrequently measured quantities (e.g., temperature). In such applications, network bandwidth is not the biggest bottleneck. Instead, approaches that save energy (e.g., by turning nodes off, or manipulate the length of packet preambles) are of more important concern. It is likely, however, that more data-intensive sensors will be used in the foreseeable future. For example, sensors that analyze spectral density, chemical composition, sound harmonics, or pictorial information might send longer arrays of data and invoke the communication bottleneck. Our approach is more appropriate for these higher-rate sensors. It should also be noted that our algorithm is not a replacement of existing MAC protocols, but rather a complementary functionality that can be composed with existing MAC layers. While we assign different frequencies to nodes,

those nodes on the same frequency can still use existing MAC layers to improve efficiency of communication.

While many different uses have been proposed for sensor networks, in the most common scenario, sensor nodes collect information, process it en-route (if needed), and collect the results at a set of pre-deployed base-stations. This scenario is also the most popular with deployment. Hence, optimizing for this important special case can have a large practical impact. The model assumes that sensors do not typically send packets directly to other sensors except with the purpose of passing information to or from a base-station. In a well-designed sensor network, the information flow is primarily from the sensors to the base-station, with only sporadic commands communicated in the reverse direction. We call such a model a *data collection network*. Hence, the main challenge addressed in this study is to maximize the ability of such a network to relay information to base-stations by a judicious distributed dynamic selection of communication frequencies for nodes.

Briefly, the proposed algorithm works as follows. Multiple base-stations collect data, each at a different frequency. Base-stations are assumed to be connected (e.g., via the Internet), and can therefore easily exchange data and load information externally. Hence, a sensor node does not have to be statically bound to one base-station. Instead, each sensor node independently chooses a communication frequency based on infrequently communicated global state. The independent choices of the nodes result over time in a new load distribution among the collection base-stations, that ultimately maximizes aggregate throughput.

The rest of this chapter is organized as follows. Section 7.1.1 gives an overview of related work. Section 2.2 describes the underlying analytic foundations. Section 2.3 presents the protocol design and Section 2.4 describes the implementation details of the channel selection algorithm on the MicaZ motes. Section 2.5 presents an evaluation both on a real testbed and in simulation. While the former is needed to provide experimental validation, the latter is important in that it explores a larger space of network design parameters. The chapter

concludes with Section 2.6, which presents a summary and directions for future work.

2.2 Theoretical Foundations

To maximize the throughput in a multi-channel network during high load, it is necessary to balance the load at the different base stations. Otherwise, congestion at the over-loaded channels will reduce the total network throughput. This section describes distributed control algorithms to achieve load balancing in multi-channel sensor networks. We consider a sensor network with N nodes and C base stations. Each base station communicates on a dedicated channel, and each node is connected to only one base station at a time (possibly through a number of hops). The load and number of nodes at channel i are denoted M_i and N_i, respectively.

To balance the network load, each node periodically decides which channel to use based on measurements coming from its current base station. We assume that all base stations are connected and thereby are able to share their load measurements. Thus, each node receives, at each sampling instant, a packet containing the vector $M = [M_1 M_2 \dots M_C]$. The packet also holds the current value of N, the total number of nodes in the network. The control signals computed in each node at every sampling instance are probabilities for that node to switch to another channel than the one it is currently using. The probability for a node at channel i to switch to another channel, j, is denoted P_{ij}.

The switch probabilities are local control signals in the nodes, designed to control the global performance of the distributed load balancing problem. The reason for using switch probabilities is to avoid excessive oscillations in the loads at the different channels. It is only desirable that a fraction of the nodes in the overloaded channels switch in order to balance the network. Furthermore, in networks of large diameter, or if the sampling interval of the switch decisions is short, the load measurements received at the nodes may experience substantial delay in relation to the sampling interval. This delay may cause instabilities in the control loop manifested by oscillation in the channel loads.

In the following sections we model the system, and use this model to come up with appropriate control strategies to achieve load balancing. We also show how these strategies may induce heavy oscillations in the network loads as a result of non-trivial delay in the measurements. It is finally demonstrated how these oscillations are eliminated by adjusting the controller gain based on a dynamic analysis of the closed-loop control system.

2.2.1 Modeling

In the following derivation we assume that all nodes in the network contribute equally with a load denoted by L. Furthermore, the equations below assume that all nodes are able to switch, and in doing so they only bring their own load to the new channel. The larger load at parent nodes will later be considered as an additional scaling factor on the switch probabilities.

The load at channel i at time k is given as $M_i(k) = N_i(k) \cdot L$. The switch probabilities $P_{ij}(k)$ computed at time k, will lead to a fraction of the $N_i(k)$ nodes leaving and other nodes joining from other channels. We have at time $k+1$ (the next sampling interval)

$$N_i(k+1) \cdot L = \left(1 - \sum_{j \neq i} P_{ij}(k)\right) \cdot N_i(k) \cdot L + \sum_{j \neq i} (P_{ji}(k) \cdot N_j(k)) \cdot L, \tag{2.1}$$

or expressed in terms of the channel loads

$$M_i(k+1) = M_i(k) - \sum_{j \neq i} (P_{ij}(k) M_i(k)) + \sum_{j \neq i} (P_{ji}(k) M_j(k)) \tag{2.2}$$

2.2.2 Static Control

The first approach will not consider the possible delay arising in the control loop, and compute the switch probabilities as functions of the current load measurements, i.e., $P_{ij}(k) = f(M_1(k), \ldots, M_C(k))$. In the following we denote by M_r, the average load at all channels,

$M_r = \frac{\sum_{i=1}^{C} M_i}{C}$. The desired stationary solution is $M_i(k+1) = M_r \ \forall i$ and $P_{ij}(k+1) = 0 \ \forall i \neq j$, i.e., we achieve the stationary solution in one sample.

There are infinitely many ways to choose the switch probabilities, $P_{ij}(k)$, to achieve the stationary solution. However, to minimize the total number of switches, we present a control scheme in which channels, i, for which $M_i(k) > M_r$ distribute their excess load between the underloaded channels and nodes in the underloaded channels do not switch. Furthermore, the switch probability will be higher to switch to the more underloaded channels.

Define S as the set of channels for which $M_i(k) < M_r$, i.e., the underloaded channels. The algorithm then becomes

$$\begin{aligned} P_{ij}(k) &= 0 \ \forall j \ \text{ if } M_i(k) \leq M_r, \\ P_{ij}(k) &= 0 \text{ if } M_i(k) > M_r \text{ and } j \notin S, \\ P_{ij}(k) &= \frac{M_i(k) - M_r}{k_j \cdot M_i(k)} \text{ if } M_i(k) > M_r \text{ and } j \in S \\ k_j &= \frac{\sum_{n \notin S}(M_n(k) - M_r)}{M_r - M_j(k)} \end{aligned} \tag{2.3}$$

The first equation specifies that the switch probabilities for nodes in the underloaded channels should be zero. The second equation says that no switching should occur between overloaded channels. Finally, the last equation gives the switch probabilities from overloaded to underloaded channels to achieve that $M_i(k+1) = M_r \ \forall i$. When all loads are equal, the switch probabilities will be zero.

To show that the static control given by Equation (2.3) achieves balanced load $M_i(k+1) = M_r \ \forall i$, we substitute the probabilities in the model (2.2). For overloaded channels, i, we

have that $P_{ji}(k) = 0$, which gives

$$
\begin{aligned}
M_i(k+1) = M_i(k) - \sum_{j \in S}(P_{ij}(k) \cdot M_i(k)) = \\
M_i(k) - \sum_{j \in S}(\frac{M_i(k) - M_r}{k_j}) = \\
M_i(k) - \sum_{j \in S}(\frac{(M_i(k) - M_r) \cdot (M_r - M_j(k))}{\sum_{n \notin S}(M_n(k) - M_r)}) = \\
M_i(k) - (M_i(k) - M_r) \cdot \frac{\sum_{j \in S}(M_r - M_j(k))}{\sum_{n \notin S}(M_n(k) - M_r)} \\
= M_i(k) - M_i(k) + M_r = M_r
\end{aligned}
\tag{2.4}
$$

A similar derivation can be performed to show that $M_i(k+1) = M_r$ also for the underloaded channels (here $P_{ij}(k) = 0$).

The switch probability derivation above has assumed that all nodes contribute equally and that each node only brings it own load if it decides to switch. This is not true for nodes closer to the base stations, which will also bring the load of their children if they decide to switch channel. It may also be the case that the loads of individual nodes are different and change dynamically. To deal with this fact, we will add a scaling to the probabilities in Equation (2.3).

Since the control packets from the base stations contain the total number of nodes in the network, N, the average load per node may be estimated as $\hat{L} = \frac{\sum_{i=1}^{C} M_i}{N}$. The scaling is then computed as $\hat{L}/\bar{L}$, where $\bar{L}$ is the total load at the node. If all nodes contribute equally and all are one hop from the base station, then those scaling factors will all be 1. On the other hand, if a subtree contains, e.g., four nodes (parent and three children), then the probability for the parent to switch will be 1/4 of that of the children. This scaling of the probabilities will effectively reduce the likelihood of large fluctuations in the network load as a result of parent nodes (close to the base stations) switching channels. This will also be evaluated in

simulation in Section 3.5.2.

2.2.3 Dynamic Control

In this section we examine the effects of delayed measurements on the static control strategy developed in the previous section. It is well-known from basic control theory, see, e.g., [1], that delay decreases the stability of a feedback control system. In the presence of delay, stability may be recovered by decreasing the gain of the controller. This can be explained intuitively by the fact that, in the presence of a larger delay in a system, it takes more time for the effects of control actions to become measurable. The controller must therefore be more "patient" or react slower to perceived performance deviations in order not to "over-react". In our case, this corresponds to multiplying the switch probabilities (2.3) with constants $K(d) < 1$, where d is the input delay of the control system measured in the number of controller sampling times.

Since the model and controllers are nonlinear, an approximate analysis will be used to obtain an expression for $K(d)$. The nonlinear system model, given by Equation (2.2), has the control signals, $P_{ij}(k)$, multiplied with the current load, $M_i(k)$. When no delay is present in the system, this multiplication cancels the factors $M_i(k)$ in the denominator of the expression for the switch probabilities. By doing this cancellation also in the case of delayed measurements, where $P_{ij}(k)$ is a function of $M_i(k-d), \ldots, M_C(k-d)$, the system may be approximated as a linear integrator model

$$M_i(k+1) = M_i(k) + u_i(k), \tag{2.5}$$

where the controllers, $u_i(k)$, are P-controllers with delay, i.e., $u_i(k) = K \cdot (M_r(k-d) - M_i(k-d))$. This linear approximation works well close to the stationary solution, where the $M_i(k)$ are almost equal.

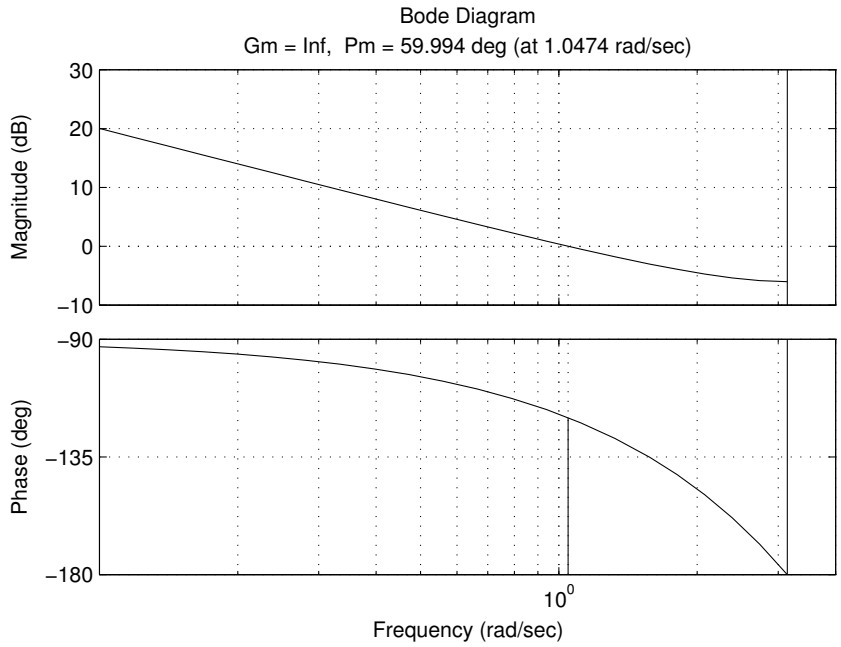


Figure 2.1: Bode diagram of the discrete-time integrator process. The phase loss induced by a time delay will be recovered by decreasing the loop gain.

We will use frequency analysis to compute $K(d)$. The condition for stability of a dynamic feedback system is that the phase of its frequency response should lie above -180 degrees at the frequency where the magnitude is equal to one. This frequency is called the cross-over frequency. The phase distance to -180 at this frequency is referred to as the phase margin. Magnitude and phase of a dynamic system are usually plotted on a logarithmic scale, which is called the *Bode diagram*.

The phase loss induced by a delay, d, at the cross-over frequency, ω_c, is equal to $\omega_c h \cdot d$, where h is the sampling interval. The Bode diagram of the system given by Equation (2.5) is shown in Figure 2.1. We can recover the phase loss induced by the delay by decreasing the gain of the system. The cross-over frequency is by definition given by $|G(e^{i\omega_c h})| = 1$, which for the integrator process (2.5) with transfer function $G(z) = \frac{K}{z-1}$ computes to $1-\cos(\omega_c h) = \frac{K^2}{2} \Rightarrow K \approx \omega_c h$. The condition for 45 degree phase margin at ω_c is

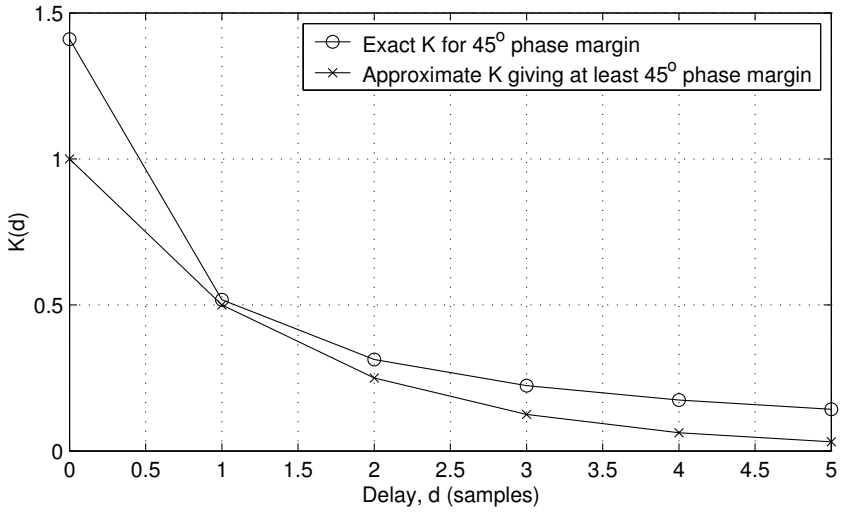


Figure 2.2: Delay-dependent controller gain, $K(d)$, to guarantee 45 degrees phase margin.

$$\begin{aligned} &\arctan\left(\frac{\cos(\omega_c h)-1}{\sin(\omega_c h)}\right) - \omega_c h \cdot d = \\ &\arctan\left(\frac{\cos(K)-1}{\sin(K)}\right) - K \cdot d = -\frac{\pi}{4} \end{aligned} \tag{2.6}$$

Figure 2.2 shows the delay-dependent gain, $K(d)$, that achieves 45 degrees phase margin for different delays. The figure also shows an approximate delay-dependent gain of $K(d) = 0.5^d$. This approximation could be used if, for example, the delay changes dynamically, and an adaptive delay compensation should be applied. This will simplify computation of $K(d)$ and still guarantee a phase margin of at least 45 degrees for all d. The phase margin will increase somewhat for larger delays, but the simple rule to decrease the loop gain a factor 2 for each sample delay will work fine for reasonable values of the delay. If the delay is very large, this simple rule will render the system unnecessarily slow.

As an example, we will consider the case of two base stations. Here, the nonlinear model equations become

$$\begin{aligned} M_1(k+1) &= M_1(k) - P_{12}(k) \cdot M_1(k) + P_{21}(k) \cdot M_2(k) \\ M_2(k+1) &= M_2(k) + P_{12}(k) \cdot M_1(k) - P_{21}(k) \cdot M_2(k), \end{aligned} \tag{2.7}$$

with the delayed control

$$
\begin{aligned}
P_{12}(k) &= \left(\frac{M_1(k-d) - M_r(k-d)}{M_1(k-d)} \right) \text{ if } M_1(k-d) > M_r(k-d) \\
P_{12}(k) &= 0 \text{ otherwise} \\
& \\
P_{21}(k) &= \left(\frac{M_2(k-d) - M_r(k-d)}{M_2(k-d)} \right) \text{ if } M_2(k-d) > M_r(k-d) \\
P_{21}(k) &= 0 \text{ otherwise}
\end{aligned}
\tag{2.8}
$$

Figure 2.3 shows simulations of the system given by Equation (2.7) under the control given by Equation (2.8) with and without delay compensation. Observe that we simulate the original nonlinear system model (and not the linearized one) together with the controller derived from system linearization. The simulation considers 48 nodes all contributing equally (load per node = 1). At time zero the system is unbalanced, with $M_1(0) = 10$ and $M_2(0) = 38$. The objective is to bring both loads to $M_r = 24$ by adapting the data collection trees.

The top two graphs show the performance for $d = 1$ and $d = 2$ samples, respectively, for uncompensated control. The destabilizing effect of the delay is clearly seen. The bottom two plots show the improvements that are achieved by adjusting the controller according to the derivation in this section. This demonstrates that the design above, which was based on a linear approximation, also works when applied to the nonlinear system model. The delay compensation will be evaluated in Sections 2.5.1 and 3.5.2, against real and simulated networks.

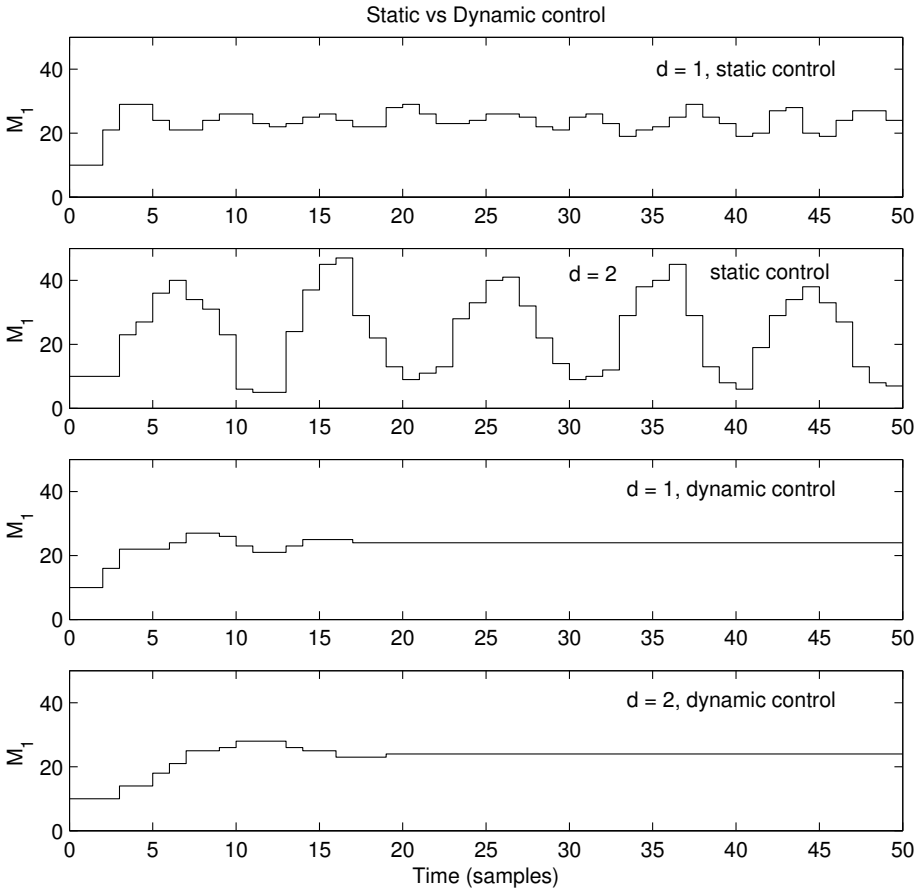


Figure 2.3: Simulation of the nonlinear load model with and without delay-compensated control. The plots show the load at channel 1 when controlled from the initial condition $M_1(0) = 10$. The delay compensation removes the oscillations.

2.3 Design

In this section, we describe the design of a network protocol used to implement the dynamic channel selection strategy. We present the protocol in such a way that programmers who are used to an event-driven programming paradigm will feel familiar and be able to re-implement it. The five types of messages used in the protocol are shown in the table below.

The design presented here is for both collection and aggregation protocols. Most of the design is similar for both protocols. The differences are in the updating of the load at the nodes and base-stations, which will be presented separately at the end of this section.

Message Type	Included Information
CONTROL	Load on all channels, sequence number, hop of sender, node count
JOIN REQUEST	sender ID
JOIN ACCEPT	hop of sender
LEAVE ANNOUNCEMENT	New channel
DATA	Data

Summary of the five types of messages.

2.3.1 Channel Load Consensus Among Nodes

As mentioned in Section 2.2, the loads of all channels are needed for the channel allocation algorithm. This information is included in messages called CONTROL messages sent from the base stations. CONTROL messages are broadcast out periodically from the base-stations every T time units. When a node receives a CONTROL message, it makes sure that the message is not redundant by checking its sequence number. The node then follows the

channel selection algorithm to select a new channel to operate on until a new CONTROL message arrives. If the result of the selection is the existing channel, the node rebroadcasts the CONTROL message for neighboring nodes.

2.3.2 Two Cases of Channel Selection

When a node switches to a new channel based on the designed strategy, we call that on-policy channel selection. Before a node makes an on-policy channel selection, it broadcasts a message notifying neighbors that it is leaving. This message is called a LEAVE ANNOUNCEMENT message. It then switches to the new channel without re-broadcasting the CONTROL message. Upon reception of a LEAVE ANNOUNCEMENT message at a child, that child will also switch to the same channel after re-broadcasting its own LEAVE ANNOUNCEMENT message. This makes all nodes under a sub-tree follow their parent to the new channel.

On-policy channel selection is not always successful, since there may be no reachable nodes around on the new channel. When that happens, the node has to switch to another channel with the hope that it can locate a parent. We call this off-policy channel selection.

The idea of the off-policy channel selection is simple: a node will try randomly a new channel which it has not used since the last failed on-policy selection. This can be implemented by a ring in which each position represents a channel. At any instance of time, there is a pointer pointing to the current channel. Whenever a node executes an off-policy selection, the pointer is shifted clock-wise. The order of channels in the ring is shuffled randomly when the node is turned on. This mechanism guarantees, after each switching, that the probability to successfully secure a parent is increased (with the assumption that the density of nodes in the network is relatively even). Off-policy channel selection also happens when a node fails repeatedly to send out data messages to its parent node.

2.3.3 Parent Selection

Once a new channel is selected, a node needs to choose a parent out of those neighbors on the new channel. Parent selection occurs in two cases.

The first case is when a node first arrives at a channel. It then broadcasts out a JOIN REQUEST message. If any node gets a JOIN REQUEST message and if it already has a parent, it will reply back with a JOIN ACCEPT message to "adopt" the requesting node as a child. As soon as the child gets a JOIN ACCEPT message, it will select the sender as parent and start to forward DATA messages from its buffer immediately. Eventually, the child may get other JOIN ACCEPT messages back. Whenever it gets a JOIN ACCEPT message, while it has a parent already, it compares the hop of the sender and the signal strength with the ones of the current parent. If the sender has a better pair of {hop, signal strength} – fewer hops and stronger signal strength are preferred – it selects that sender as its new parent. The smallest hop is preferred to make sure that there will be no circle in the tree, and strong signal strength is preferred since it is a good and cheap indicator of link quality [2].

In the case the node does not get back any JOIN ACCEPT messages after a specific window of time, it will make an off-policy channel selection. By selecting the first responder as parent immediately and then considering new parent candidates when new JOIN ACCEPT messages arrive, a node can start to send out data messages at the earliest time, yet still eventually choose the most preferred parent.

The second case of parent selection is when a node selects a new parent at the time of receiving a CONTROL message. It again compares the pair {hop, signal strength} of its existing parent with the sender of the CONTROL message. The node will choose the sender as its new parent if the sender has a better pair. This process helps the node maintain a reasonably good parent in spite of the variance of the wireless links. Since parent changing is done by simply changing a variable addressing the receiver of out-going data messages, it can be done in negligible time.

2.3.4 Updating Local Load

The local load at each node is part of the information needed to select a channel when a new CONTROL message arrives. It means that we need to measure it every T time units. There are differences in notations of load and how to calculate it between collection and aggregation protocols.

2.3.4.1 Data Collection Protocol

In the collection protocol, we define the load of a channel as the total number of data packets arriving per sample at the base-station dedicated for that channel. Consequently, local load at a node is the total number of data packets received from its descendants plus the number of data packets generated by itself. The load at a channel is simply computed as the arrived data-packets per time unit at the corresponding base-station.

To calculate the load at a node, a trivial solution is to count the number of arrived and generated messages of a node every T time units. However, load can be updated after the arrival of a CONTROL message. In that case, the load used for channel selection is out-of-date and will affect the channel selection policy. To avoid that, we use a buffer ring which contains cumulative load of a node at the end of each shorter period. The total length of these periods is equal to T. When a CONTROL message arrives, load is computed simply by subtracting the tail entry of the buffer from the head entry of the buffer. By doing that, we have a more up-to-date estimation of the current local load.

2.3.4.2 Data Aggregation Protocol

In the aggregation protocol, data messages can be aggregated with others. Therefore, counting the number of arrived messages at a basestation does not make sense anymore. Instead, we define the load of a channel as the total number of data messages transmitted in the

frequency of that channel. With that definition, the load balancing effort still contributes to maximizing the total throughput of the network. Consequently, the load at a node is defined as the total number of data messages transmitted by its descendants and itself within a period of time.

To be able to compute loads, we introduce a counter in the header of each data message. When a data message is first generated, its counter is one. When messages are aggregated to create a new message, the counter of the new message is the sum of all counters of the contributing messages plus one. The load at each node is now computed as the sum of the counters of all received messages plus the total number of data messages sent out. Finally, the load M_i at a channel is the sum of the counters in the received data messages at the corresponding base-station during one sample.

2.4 Implementation

There are three different elements to our protocol: nodes, base-stations, and a distributed coordinator. The task of the nodes is to collect data and forward data from their children to their parent. Base-stations can be considered as a special type of node which does not collect data but only receives data messages from its children. The network can be seen as a collection of trees rooted at the base-stations. Each base-station works on a different frequency. The coordinator exchanges load information among all base-stations (e.g., via their Internet connection) who then gather CONTROL messages and send them back to the sensor network. In our implementation, the nodes are MicaZ motes and the base-stations are workstations with MicaZ motes interfaced via MIB510 boards. The coordinator is a Java program running on the workstations.

Figure 2.4 precisely defines the behavior of the nodes. The circles are states without actions and are reached from an unconditional transition from other states. The rounded boxes are states associated with an action. The solid arrows are events. The dotted arrows are unconditional transitions from one state to another state. These are used to conveniently describe actions with multiple steps.

The compiled code for the general protocol which is applicable for both collection and aggregation protocol is 13542 bytes in ROM and 1091 bytes in RAM.

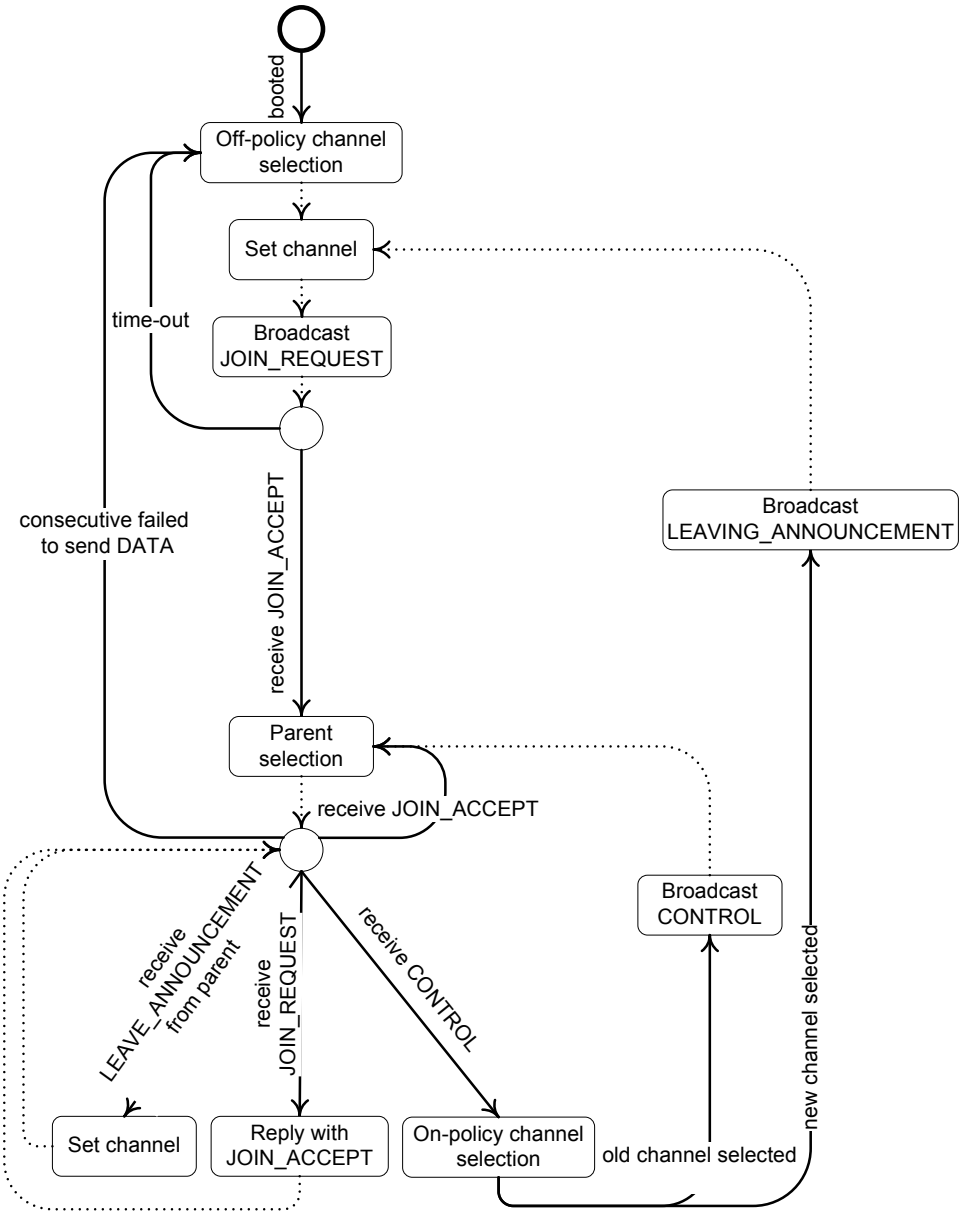


Figure 2.4: State machine for sensor nodes.

2.5 Evaluation

This section will present evaluation results for both experiments on a real testbed and simulations. Two separate sets of evaluations have been conducted to show different aspects of the proposed control-theoretic load balancing strategy. The first will show that the control strategies proposed in Section 2.2 actually achieve load balancing in the network. Here it will also be demonstrated that the dynamic control strategy is effective in reducing oscillations when non-trivial delay exists in the load measurements. The second evaluation set will show the improvements in network throughput that can be obtained using the suggested strategy.

2.5.1 Evaluation on an Experimental Testbed

2.5.1.1 Load Balancing

The first testbed experiment was done with 4 channels, and a frequency distance between the channels of 2 MHz. 4 base stations were used and each used a separate channel to collect data from a network of 24 nodes. The loads of all channels were sampled and broadcast with CONTROL messages from the base stations every $T = 2$ seconds. Each node generated data with the same rate of 10 messages/sample. Figure 2.5 shows load balancing at channel 1 with compensation for different input delays, d. As the graph shows, the load balancing works best for a compensation for $d = 3 - 4$.

The second experiment was done with a similar setting, except that the number of nodes was increased to 36. As shown in Figure 2.6, the load on each channel now is not really a straight line after the transient as was the case for 24 nodes. The reason is that when the size of the network increases, the chance that messages get lost or are being dropped also increases. That makes the load less smooth than with 24 nodes. We also examined the average error rate w.r.t. the input delay d and realized that the best delay compensation for

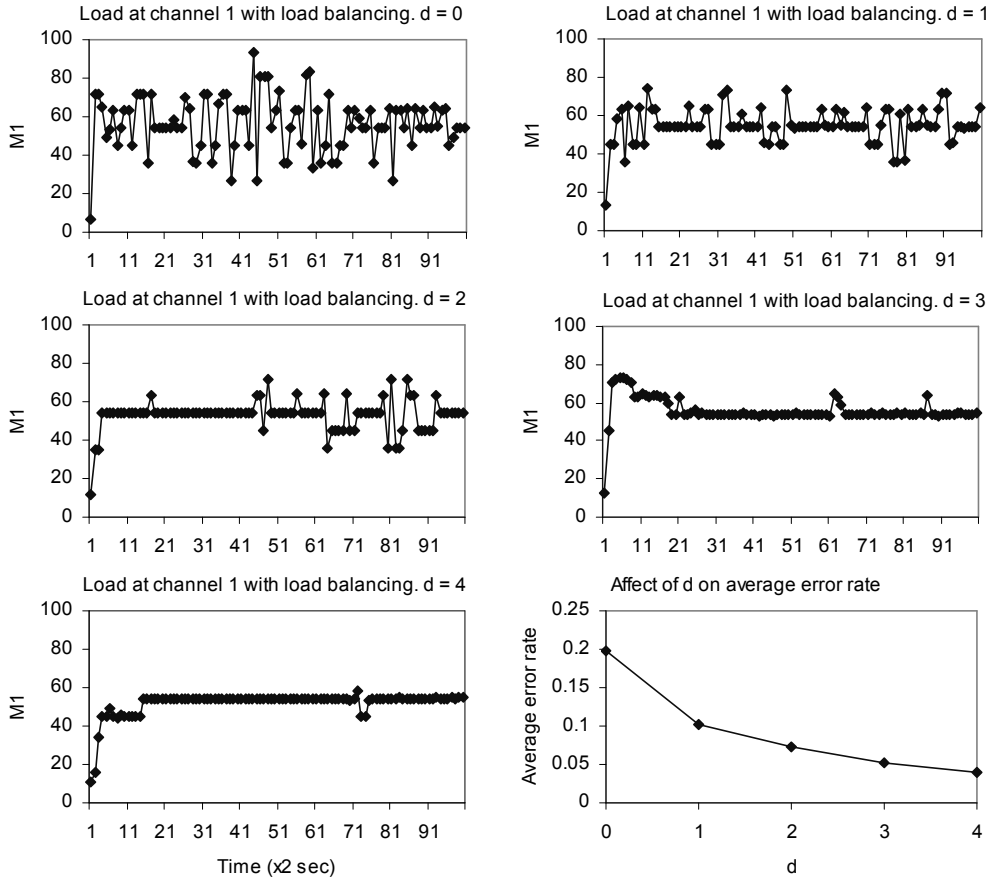


Figure 2.5: Load balancing experiment on a testbed with 4 channels, 4 base stations and 24 nodes.

the network now is 7, which cause the least average error ratio in the load balancing for the network. The error ratio was computed as $|M_i - M_r|/M_r$. When the delay compensation increases to 8, it takes longer time for the load to get balanced, which makes the average error ratio during the experiment higher than for $d = 7$. The optimal value of d simply expresses the actual delay in the control loop as a multiple of sampling time. Later, we show how delay can be determined dynamically by our protocol so that it tunes itself automatically.

A third load balancing experiment was done with 2 channels, 2 base stations, and 24 nodes. The nodes in the network were divided into two logical groups: A and B. When the network first started, nodes in A produced data messages with the rate 10 messages/sample and nodes in B produced with double data rate at 20 messages/sample. After every 40

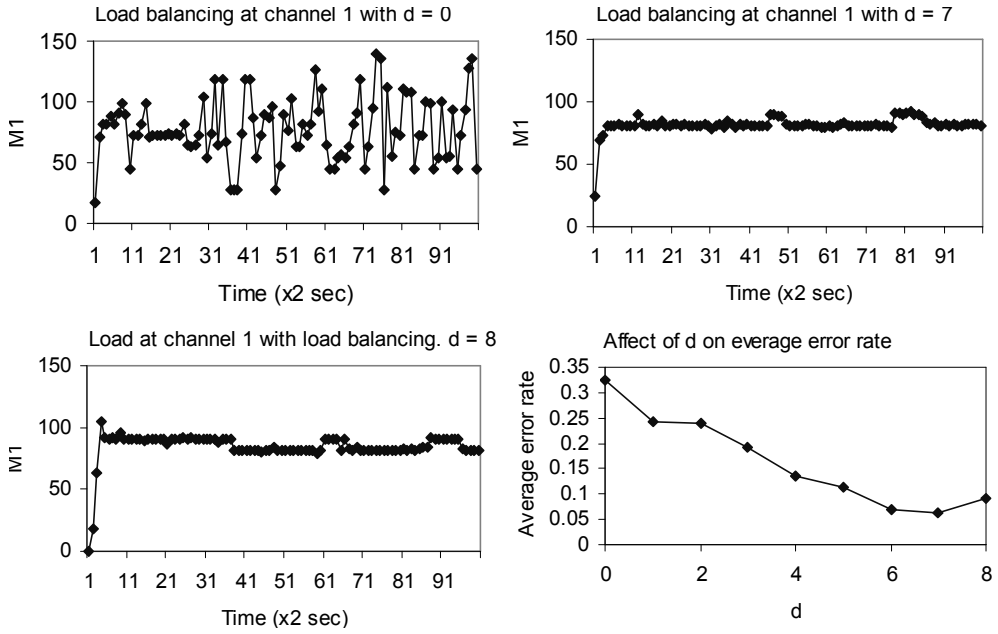


Figure 2.6: Load balancing experiment on a test-bed with 4 channels, 4 base stations and 36 nodes.

samples, the roles were reversed. Figure 2.7 shows the load ratio of channel 1 over 1000 samples. As we predicted, the uneven data-rate of the nodes was handled well by the load balancing algorithm. There was an interesting observation that at the first 4 data-rate changing periods, the oscillations caused by the sudden data-rate change is noticeable. After that, it becomes indistinguishable from the random noise in the load measurements. We observe that nodes in the network were distributed naturally in a way such that on each channel, there were approximately half of the nodes from A and half of the nodes from B. In that configuration, even as the data-rate changes every 40 samples, the total load on each channel is still almost the same.

The fourth experiment investigates adaptive delay compensation for a data aggregation network. In the first experiment in this section we saw that the delay arising in the control loop will depend on the number of nodes. Furthermore, the delay experienced by the different nodes may vary over time. For this reason it may not be possible to decide a proper value of d off-line. If it is chosen too small, we would expect large oscillations at high load. However,

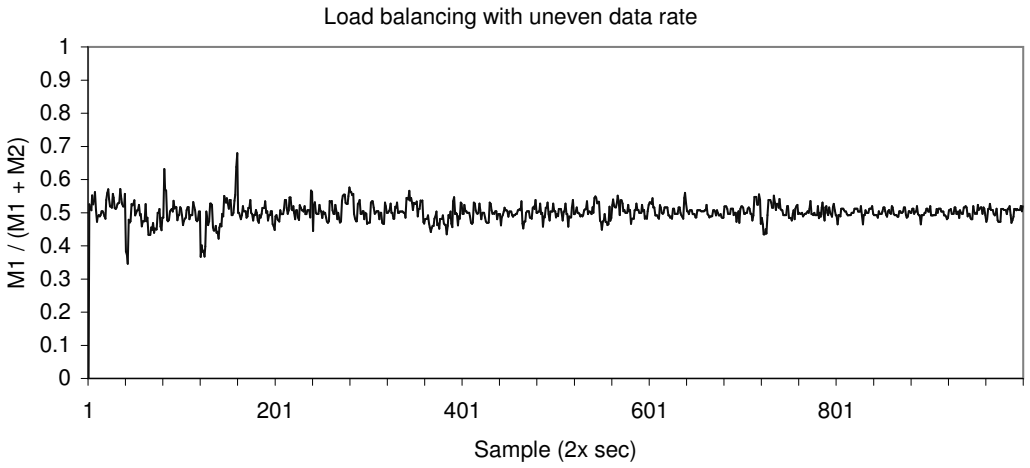


Figure 2.7: Load balancing experiment on a testbed with 2 channels, 2 base stations and 24 nodes. The nodes are divided in two groups A and B. At first, the data-rate at A is twice that of B. After every 40 samples, all nodes change group.

if we choose it too large, the control would be unnecessarily slow when the actual delay is low. Instead, if the real delay of the network can be measured and known by all nodes, the optimal error rate can be achieved without a-priori knowledge of d.

This leads to an adaptive solution, in which the base stations collaborate to figure out the real network delay and include that information in CONTROL messages. Figure 2.8 shows the performance when applying an adaptive delay compensation scheme. The experiment was done with 2 channels, 2 base-stations, and 36 nodes. In the first 100 samples, 18 nodes are present in the network, half of them have double the rate of the other half. Another group of 18 nodes later joins the network, half of them also have double the rate of the other half. After the 100th sample, the two halves of the first group switch their data rates. As we observe, it takes around 10 samples for the adaptive scheme to balance the load after both changes. No serious oscillations are present in the load.

Finally, we will present how the scheme estimates the delay on-line. It is non-trivial to measure a common delay in the network, since each tree may have different delays. Instead, each node will maintain a sequence number to be embedded in each new data message. Initially, that sequence number is initiated as the sequence number of the last received

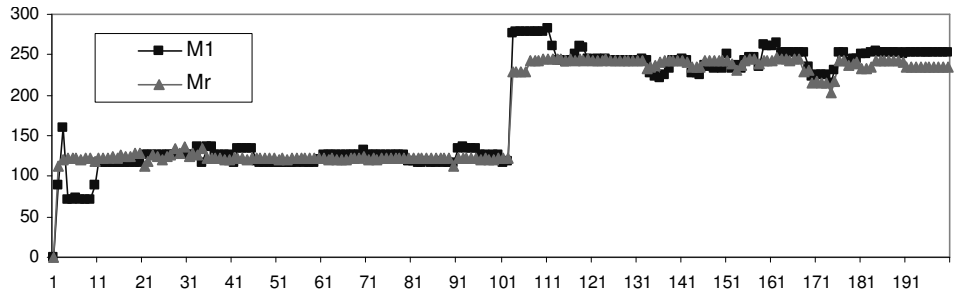


Figure 2.8: Load balancing with adaptive input delay compensation for aggregation protocol: M_r is the ideal load on the channels with perfect balance, M_1 is the load on channel 1.

CONTROL message. Each time a node gets a packet from its children, it compares the sequence number in the received packet with its sequence number, and resets its sequence number with the sequence number in the received packet if that number is smaller. Anytime a packet is generated at the node, that current sequence number will be attached to its header. These sequence numbers are unchanged when data packets are being forwarded to base-stations. Every T time units, the sample period, base-stations include the smallest sequence number from data packets received within that period to the load report before sending it to the coordinator. The coordinator then selects the smallest sequence number among the channels, and the difference between that number and the sequence number of the coordinator is used as an estimate of the delay.

The above algorithm estimates loop delay. The base-station, in effect, measures how many sampling intervals it took for its CONTROL message to be diffused down the tree and for packets from nodes receiving that message to get back to the root. However, if this raw estimate of delay is embedded immediately into the CONTROL message, actual delay may be understimated because those packets experiencing the largest delay will be received last. Delay under-estimation may cause large oscillations as the loop becomes less stable. To handle this case, the raw delay samples are considered within a sliding time window. At any instance, the delay fed to the CONTROL message is the maximum delay in that window. The

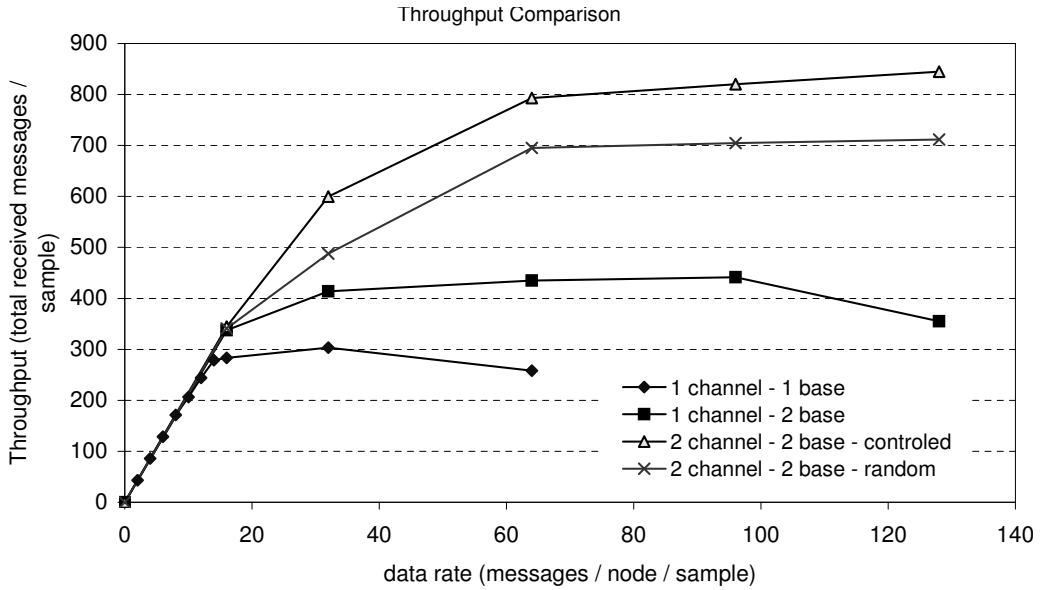


Figure 2.9: Throughput comparison in a 24-node network for: i) 1 channel, 1 base stations; ii) 1 channel, 2 base stations; iii) 2 channels, 2 base stations with random channel selection; iv) 2 channels, 2 base stations with load-balancing channel selection.

size of the sliding window itself should be of the order of the worst case possible network delay so that delay estimates from the slowest packets can be considered.

2.5.1.2 Throughput Comparison

In this subset of experiments, we compare the throughput of the network with 24 nodes for different channel settings. We consider 4 cases: i) network with 1 channel, 1 base station; ii) network with 1 channel, 2 base stations; iii) network with 2 channels, 2 base stations with random channel selection strategy; and iv) network with 2 channels, 2 base stations with load-balancing channel selection strategy. The idea is to separate throughput improvements due to use of multiple channels from that due to adaptive tree load-balancing. The nodes have uneven data-rate as described in the third experiment in the previous section. The results are shown in Figure 2.9, in which the the x-axis shows the average data-rate of the nodes. The y-axis shows the throughput in received packets/sample.

We see that all cases get the same ideal throughput when the data-rate does not exceed

16 messages/sample. When the data-rate increases, the throughput of the 1 channel, 1 base station network drops and decreases after 32 messages/sample when collisions become serious. The network with 1 channel, 2 base stations works better, since it has less of a bottleneck and, hence, the packet drops at the base-stations are reduced. Furthermore, the base stations were positioned at two opposite sides of the network to maximize spatial re-use. The third plot shows that the case with 2 channels and 2 base stations out-performs both cases of 1 channel when the data-rate exceeds 16 messages/sample. Finally, from the fourth plot, the throughput of the network with load-balancing is better than the network with random channel selection when the data rate goes beyond 32 message/sample. Thus, the tree load-balancing algorithm is needed to best utilize the available channel diversity.

2.5.2 Simulation and Scaling

Simulations were conducted to complement the experiments and to evaluate the strategy for a larger network and with other parameters.

2.5.2.1 Simulation Environment and Setup

The simulations have been run using Matlab and the TrueTime [3] toolbox. This simulator uses Simulink, the graphical simulation environment of Matlab, and provides simulation blocks to model networks and computer nodes. Each computer node contains arbitrary code (tasks and interrupt handlers) written by the user as Matlab functions. The network blocks simulate common wired and wireless MAC layer policies as well as a radio model for simulation of wireless transmissions.

Each mote has an adjustable signal power and can communicate on a number of channels. The transmission power and the receiver threshold are adjusted to change the network diameter, i.e., to experiment with different number of hops from the different nodes to the base stations. Furthermore, it is possible to change other network parameters, such as the data rate, minimum frame size of packets, acknowledgment time-outs, and the maximum number of retransmissions of un-acknowledged packets.

In the following simulations we will simulate the ZigBee protocol with a data rate of 250 kbit/second. The packets sent by the nodes are of size 40 bytes. The simulations consider a setup of 48 nodes arranged in a 6 by 8 grid. Of these nodes, two are base stations communicating on two different channels.

2.5.2.2 Load Balancing

This simulation will investigate adaptive delay compensation as described in the experimental evaluation. We again consider two network halves with different load. At the middle of the

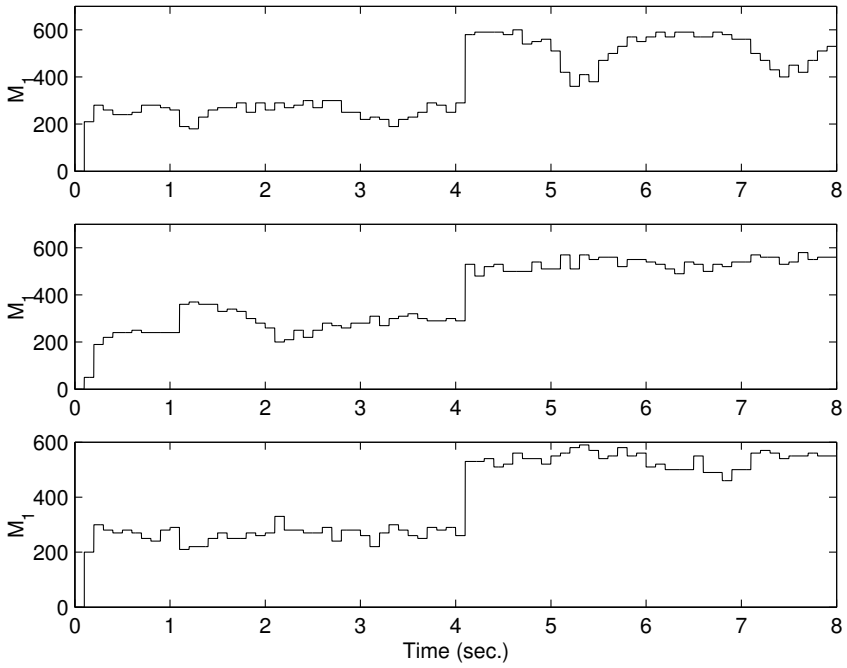


Figure 2.10: Simulation of the adaptive delay compensation. The top plot compensates for a too short delay, thus the oscillations at high load. The middle plot shows static compensation for the large delay, which makes the system too slow in the beginning when the delay is short.

simulation run, the total load increases and the two network halves change. This will cause a transient in the load balancing and increased delay in the loop.

The results are shown in Figure 2.10. The top and middle plots show static compensations for $d = 1$ and $d = 3$, respectively. Static compensation for $d = 1$ works well in the beginning when the actual delay is small, whereas we get an oscillatory behavior when the true delay increases. On the other hand, the static compensation for $d = 3$ works well in the end, but gives a slow transient response in the beginning. The bottom plot shows the improved performance achieved by an adaptive delay compensation, where the actual loop delay is estimated and sent to the nodes in the CONTROL messages. The average of the absolute normalized error was computed to 0.11, 0.078, and 0.053 for the respective cases.

The simulations and experiments have, thus, shown that the approximate dynamic control strategy derived in Section 2.2 is effective to avoid oscillations in the load in the real and

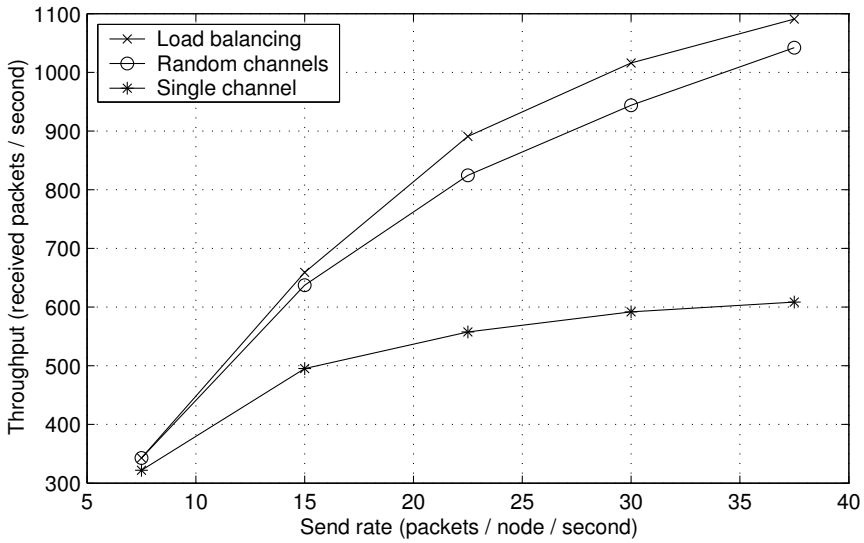


Figure 2.11: Comparison of throughput between the proposed load balancing scheme and random channel selection in the two-channel case and 48 nodes. The figure also shows the throughput when using a single channel.

simulated networks. Furthermore, if the delay changes dynamically this could be accounted for on-line in order to achieve the best possible performance.

2.5.2.3 Throughput and Probability Scaling

The first simulation will investigate the throughput improvement achieved by our strategy in the two-channel scenario with 48 nodes. As in the experimental evaluation, we compare against a baseline case in which each node just randomly chooses a channel which it then remains at. The results are shown in Figure 2.11. The x-axis shows the average send rate of the nodes and the y-axis displays the number of received packets per second. An improvement of around 10 percent is observed. The graph also shows the throughput for the case when all nodes use the same channel and send data to the closest of two base-stations. As expected, a substantial performance improvement can be seen between the one- and two-channel cases. The results correspond well to the experiments.

The next simulation studies the throughput as a function of the network diameter mea-

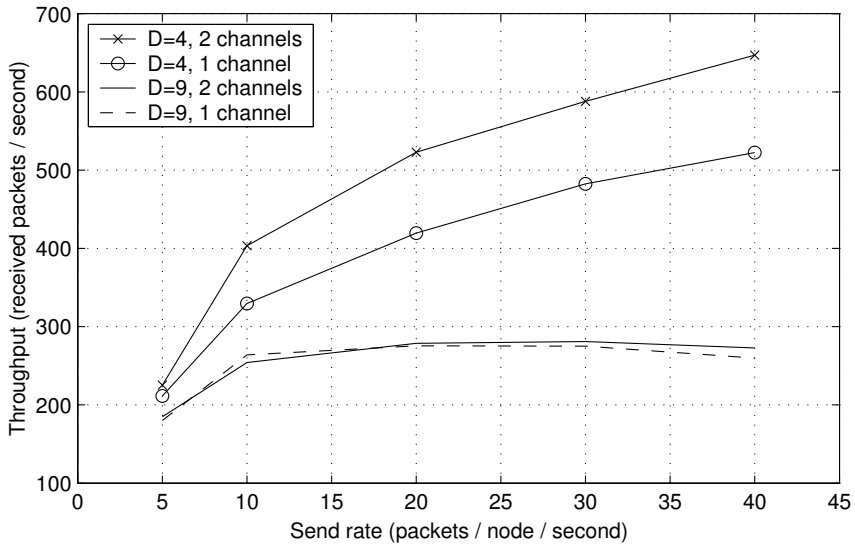


Figure 2.12: Throughput comparison between one and two channels for networks with different diameter D (measured in hops).

sured in hops. The results are shown in Figure 2.12, comparing 4 and 9 hops for one and two channels, respectively. In the one-channel case, the network is divided in two halves, each sending to one base station. We observe that the throughput gained by using two channels is much higher in the 4-hop case. This is due to the reduced interference between nodes in larger diameter networks, where nodes only interfere with its closest neighbors. Thus, the expected gain by the multi-channel strategy is more profound for networks with fewer number of hops. For reliability, latency, and energy balancing considerations, it is expected that practical data collection sensor networks will have a sufficiently small diameter, which is the case where the presented algorithm helps most.

Finally, the 4-hop simulation setup was used to evaluate the influence of the switch probability scaling outlined in Section 2.2.2. Figure 2.13 shows the switch frequency as a function of the distance (in hops) to the base station. Observe that, as a result of the scaling, the switch frequency is much higher for nodes farther from the base stations. Consequently, pro-longed oscillations due to parent node switching are very unlikely and will be balanced by switching occurring in the children nodes.

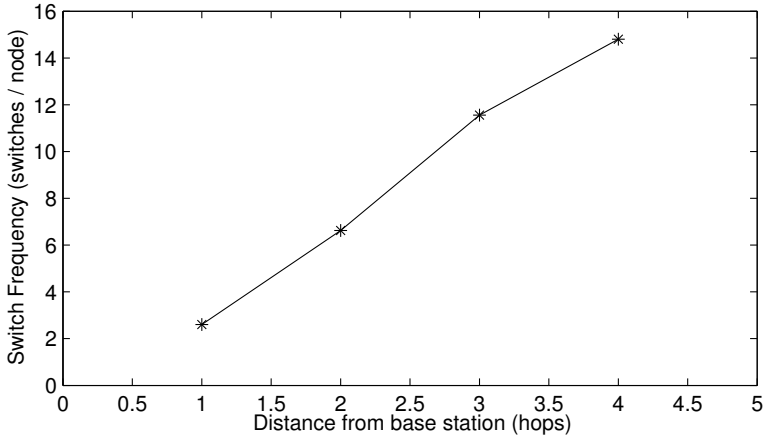


Figure 2.13: Switch frequency as a function of the distance (in hops) to the base station.

2.6 Conclusions

This chapter has presented a control-theoretic strategy to maximize throughput in multi-channel sensor networks. The problem was cast as one of distributed load balancing, in which nodes periodically take decisions on which channel to use. It was shown that a dynamic control design was needed to avoid oscillations in the network load due to delays arising in the control loop. Stability conditions were derived and analyzed. Design and implementation of a channel selection protocol were presented for both data collection and aggregation networks. The proposed strategy was evaluated both on an experimental testbed and in detailed simulation. The next chapter will consider the design and implementation of MAC layer protocols tailored to multi-channel sensor networks.

CHAPTER 3

A PRACTICAL MULTI-CHANNEL MEDIA ACCESS CONTROL PROTOCOL FOR WIRELESS SENSOR NETWORKS

Despite availability of multiple orthogonal communication channels on common sensor network platforms, such as MicaZ motes, and despite multiple simulation-supported designs of multi-channel MAC protocols, most existing sensor networks use only one channel for communication, which is a source of bandwidth inefficiency. In this work, we design, implement, and experimentally evaluate a practical MAC protocol which utilizes multiple channels efficiently for WSNs. A control theory approach is used to dynamically allocate channels for each mote in a distributed manner transparently to the application and routing layers. The protocol assumes that sensor nodes are equipped with one half-duplex radio interface which is most common in current hardware platforms. The protocol does not require time synchronization among nodes and takes the channel switching cost of current hardware into account. Evaluation results on a real testbed show that it achieves a non-trivial bandwidth improvement using 802.15.4 radios in topologies which are typical in WSNs. The MAC protocol was implemented in TinyOS-2.x and packaged as a software component to enable seamless use with existing applications.

3.1 Introduction

This chapter presents a practical design, implementation, and evaluation of a multi-channel Media Access Control (MAC) protocol for Wireless Sensor Networks (WSNs). There has been a lot of MAC protocols introduced for WSNs that use only one channel for communication. However, with the new radio capabilities of WSN motes which can communicate on multiple frequencies, this is a great source of inefficiency. The very high density of current WSNs inevitably results in physical bandwidth limitations and heavy collisions on a single channel.

There is previous work on multi-channel MAC protocols for WSNs [4, 5, 6, 7, 8]. Some of these MAC protocols assume that the time to switch between two channels is negligible, whereas others require fine-grained time synchronization among nodes. Some assume that nodes have a multi-radio interface or can listen on different channels simultaneously. Most of these MAC protocols have only been evaluated in simulation, and the rest require devices with fully fledged multi-radio interfaces. To the best of our knowledge, all previous multi-channel protocols for sensor networks have at least one of the aforementioned limitations.

Our work is the first multi-channel MAC protocol which is implemented for MicaZ motes with only one half-duplex radio interface and with long channel switching times. In [9] a multi-channel MAC protocol was developed for collection WSNs, and was also implemented on MicaZ motes. There has been similar work for dissemination in [10] and [11]. While these efforts implemented multi-channel solutions for specific applications in sensor networks (such as data collection or dissemination), the MAC protocol described in this chapter is the first general purpose MAC protocol which is designed and implemented on sensor motes with no specific assumptions on the application.

The main idea of the protocol is to assign a *home frequency* to each node such that network throughput is maximized. We call each different frequency available to the network,

a *channel*. All nodes in the network start on the same channel. When this channel becomes overloaded, some nodes migrate to other channels to spread the communication load across non-interfering frequencies. Migration to another home frequency does not entail loss of connectivity with nodes that remain on the old home frequency. Instead, our protocol involves a mechamism whereby a node can send messages to another node that is on a different home frequency. Briefly, when a node needs to send messages to another on a different home frequency, it switches to the home channel of the destination node enough to send the message. Obviously, communication between nodes on the same home channel incurs less overhead. This motivates formulating the network throughout optimization problem in a manner similar to a clustering problem in a graph, whereby nodes that communicate frequently are clustered into the same channel, whereas those that do not communicate much (but are within each other's interference range) are separated into different channels.

Our protocol solves the problem in a distributed manner where nodes locally compute their "edge costs" and make migration decisions independently. Towards that end, nodes exchange state information about messages received and degrees of estimated communication success probability. If the estimated success probability is low, a node may switch to another channel. The switching is done based on a probability such that while alleviating congestion we avoid having all nodes jump to the new channel.

The algorithm design is reduced to two main problems. First, the graph clustering algorithm conceptually attempts to find a minimum K-way cut in the graph given by the network topology in a distributed manner in order to minimize costly inter-channel communication. Second, a configuration control problem is formulated to compute the probabilities of home channel switching. These probabilities are chosen such that the network self-configures into using just the right number of channels without excessive fluctuation among channels and without being too slow to respond to changes in load. The first problem is NP-complete, which is thus approached by an efficient distributed heuristic. For the self-configuration

problem we propose a feedback control approach to compute the switch probabilities.

Our main contribution is to show that sensor nodes with a single half-duplex radio interface can actually benefit from channel diversity. The protocol is simple and light-weight enough to be implemented on MicaZ motes. Evaluation on an actual testbed shows that it works efficiently. The experimental results in both simulation and on a real test-bed show that the new MAC protocol not only achieves higher bandwidth, but also adaptively alleviates network congestion and avoids channels with high interference due to external sources (e.g., nearby 802.11 connections). In this work, the MAC is independent from the routing layer. Power management in the presence of multiple channels is not in the scope of this work.

The implemented MAC protocol component works on top of the existing single channel MAC protocol in TinyOS-2.x and exposes the same interface as Packet Protocols in [12]. Hence, applications developed for TinyOS-2.x can be adapted to the new MAC protocol seamlessly.

The rest of this chapter is organized as follows. Section 7.1.2 gives an overview of related work. Section 3.2 describes architecture considerations and underlying analytical foundations. Section 3.3 presents the protocol design and Section 3.4 describes the implementation details of the channel selection algorithm on the MicaZ motes. Section 3.5 presents an evaluation both on a real testbed and in simulation. The testbed experiments work as proof-of-concept and are used to validate the simulation results. The simulations evaluate the MAC protocol on a larger scale. Finally, the chapter concludes with Section 3.6.

3.2 Theoretical Analysis

Our algorithm is based on three observations. First, a new channel should be allocated only when needed. If there is no serious interference or collisions in the neighborhood, nodes should not switch to other channels. This reduces the cost of inter-channel communication. Second, by design, some nodes should be more likley to initiate channel switches than others. The more global is the view (of communication) that a node has, the more informed it is, and the better equipped it is to make the right move. Third, nodes with a more limited view should act locally to minimize cross-channel communication. The best local action is to follow a node with a better view.

In this section, we present solutions and analysis to the problems of minimizing inter-channel communication and choosing the channel switching probabilities consistently with the above observations. The former is formulated as a distributed clustering problem and the latter is approached using feedback control theory. Below, we first describe our fundamental mechanism for communication between neighboring nodes. We then provide solutions to clustering and feedback control to maximize communication throughput.

Since cross-channel communication introduces extra cost both due to channel switching times and due to retransmissions caused by the deafness problem (when a node sends messages to another but uses the wrong channel), it is desirable to minimize cross-channel communication and maximize same-channel traffic. This is directly related to the objective of the K-way cut problem in graph theory and is used as an inspiration for our channel allocation protocol.

3.2.1 The K-way Cut Problem

Our approach is to partition the nodes in the network into different sets, each assigned a separate home channel, such that two types of constraints are met. First, communication

within each set is limited to local capacity. Second, communication across sets is minimized. In a graph where each node is a communication device and where link costs represent the amount of communication, this corresponds to solving a K-way cut problem of minimum K that respects the capacity constraints on each cluster. There have been several centralized approaches to solve versions of the K-way cut problem. An optimal algorithm [13] was proposed for a k-way cur with fixed K with $O(n^4)$ complexity for $K = 3$ and $O(n^9)$ complexity for $K = 4$. A more efficient algorithm [14] was subsequently proposed, which has $O(n^3)$ complexity for $K = 3$ and $O(n^4)$ complexity for $K = 4$, which is the fastest optimal algorithm for a fixed K. Some centralized approximation algorithms for undirected graphs were also developed [15]. Distributed heuristic algorithms were proposed for undirected graphs.

These algorithms require a-priori knowledge of K and are quite heavy-weight. The graphs in our problem are directed, weighted graphs without a fixed K, which makes the problem harder and more complex. Due to the constraints on the sensor devices, any algorithm with a high polynomial complexity will lack scalability.

In the following, we will describe our adaptive algorithm which takes only $O(n^2)$ computation time and $O(n)$ memory and provides reasonably good performance within the scope of our MAC protocol.

3.2.1.1 The Algorithm

With the above intuitions in mind, channels are organized as a ladder, starting with the lowest channel, F_0, up to the highest channel F_N, with N being the number of channels available in the network. Whenever a node first joins the network it starts at channel F_0 (hence, initially all nodes are the same "cluster". Once a node figures out that there are lots of messages lost due to collisions and interference (i.e., the local cluster capacity constraint is violated), the node considers switching channels. The switching decision is based on how serious the collisions and interference are and on the role of the node in contributing traffic

to the network.

To measure the effect of a crowded spectrum, each node periodically broadcasts a tuple $< s, f >$, where s is the total number of times the node successfully acquires the channel, and f is the number of times the node is unsuccessessful (at acquiring the channel). Periodically, every node i receives a set of tuples from its neighbors j. Based on that, node i estimates the probability that any of its neighbor nodes can successfully attempt to access the channel:

$$\alpha_i = \frac{\sum_j s_j}{\sum_j (s_j + f_j)}. \tag{3.1}$$

If α_i is too low, the channel must be too crowded around node i [1]. Hence, if α_i is less than a configurable value α_{ref}, node i will consider switching from its current home channel F_c to the next higher channel F_{c+1} (unless $c = N$), with a probability that depends on channel conditions. The use of switch probabilities will reduce fluctuations between channels that may otherwise result if many nodes switched channels at the same time. In the next subsection, we describe how a control scheme is used to make the channel switching process stable. For now, let's assume that given the history and current status of home channel c of node i, the node has a probability to switch from channel c to the next channel $c + 1$, denoted by $\beta^i_{c,c+1}$. In our algorithm this probability increases with the difference in quality between the source and destination channels (i.e., as the source becomes substantially worse than the destination).

In summary, as appropriate of a K-way cut heuristic (with a variable K), we operate by dividing existing clusters that exceed capacity repeatedly until capacity constraints are met. Key to the design of this heuristic is to determine the mechanism for splitting clusters and the boundaries across which splitting must occur. This reduces to two questions: who should initiate the split and who should follow into the new cluster? The solution should be

[1] Notice that α_i also reflects interference at i. If interference exists around i, node i will be able to sense the signal and cease to access the channel and that affects the value of α_i

distributed and obey the goal of minimizing communication across clusters.

To answer these questions, observe that in a wireless sensor network, nodes are usually not equal in contributing to network load. Hence, they should act differently in terms of channel switching probability. Consider two extreme examples. The first example is a node that only *sends* messages (to its neighbors) but does not receive. This pattern is consistent with that of data sources. The second example is a node that only *receives* messages (from its neighbors) but does not send. This pattern is consistent with data sinks in wireless sensor networks. Channel congestion typically occurs at sinks. Hence, sinks have a more global view of traffic than sources. As such, sinks are better positioned to make decisions on channel allocation.

In our algorithm, nodes that behave predominantly as sinks have preference to switch channels first (i.e., initiate the cluster split). This has the desirable side-effect of creating well isolated clusters. A node that acts predominantly as a sink does not send much traffic by definition, and hence has a low-cost outgoing link, making it appropriate to cut (by the K-way cut algorithm). Nodes that communicate heavily with those who switched, follow them into the new cluster. This works well for aggregation topologies, which is the predominant case in data collection networks. Finally, to communicate across clusters, a sender on one home channel simply switches to the home channel of the receiver temporarily to send messages to the latter.

More specifically, the probability that a node initiates a cluster-split by switches channels from c to $c+1$ is given by

$$P^i_{c,c+1} = MAX\left(0, sink\_factor_i \times \beta^i_{c,c+1}\right) \tag{3.2}$$

where $\beta^i_{c,c+1}$ increases with the difference in quality between channel c and $c+1$ (in favor of $c+1$), and $sink\_factor$ is an indicator showing how closely a node resembles a sink. It is

computed from

$$sink\_factor_i = \frac{In_i - Out_i}{In_i + Out_i}, \tag{3.3}$$

with In_i and Out_i being the total number of messages received and sent by i at its home channel, respectively. If the node is a true sink, $sink\_factor = 1$. If it is a pure source, $sink\_factor = -1$. An intermediate node in the network might sink some traffic and forward some. Its $sink\_factor$ will thus have some intermediate value. For example, a pure router that simply forwards all traffic will have $sink\_factor = 0$. An aggregator that summarizes the traffic and forwards the summary will have a $sink\_factor$ closer to 1.

By encouraging splitting when the current channel is much worse than the target channel, our cluster splitting mechanism guarantees that a network will allocate more channels when it gets congested hence preserving cluster capacity constraints. By letting sinks initiate the split with a higher probability, we ensure that the split starts across a low-cost link. Finally, by letting neighbors who send much traffic to those who switched follow them to the new channel, we present a natural way to grow a new cluster in a way that minimizes the communication across different clusters. We call this phase *channel expansion*. When a channel is no longer congested, nodes on this channel invite those from the next (higher) channel in the ladder to switch to the underutilized frequency. As before, sink-like nodes initiate such transitions with a higher probability. Other nodes follow. We call this phenomenon *channel shrinking*.

3.2.2 The Self-Configuration Problem

The self-configuration problem considers the dynamics of channel expansion and channel shrinking. In particular, it is important that such transitions are stable. Otherwise, nodes may incur a significant overhead switching between channels. Both the channel expansion and channel shrinking mechanisms are designed using feedback control theory, where the

control signal is the probability for a node to switch channel. The use of probabilities takes the distributed nature of the control system into account and prevents all nodes from switching at the same time, which would not improve the situation.

The control laws for channel expansion and channel shrinking have been designed to be intuitive and easy to implement. However, we will also present an analysis that shows how to choose the controller gain parameters to achieve a good trade-off between fluctuations and performance. The analysis is based on restricting the fraction of nodes that are allowed to switch channel during a certain time interval related to the time delay in the system. The restriction ensures that the loop is stable in a control-theoretic sense.

3.2.2.1 Channel Expansion

We propose the following feedback control scheme for the channel expansion. The probability for a node i to switch from its current channel c to the next channel $c+1$ if $\alpha_c^i < \alpha_{ref}^{up}$ is computed as

$$\beta_{c,c+1}^i(k) = \beta_{c,c+1}^i(k-1) + K_r^{up}\left(\alpha_{ref}^{up} - \alpha_c^i(k)\right), \tag{3.4}$$

where k denotes the sampling interval (i.e., the time between consecutive updates of the switch probabilities). The controller is in integral form (i.e., its output is proportional to the integral of inputs), where the switch probability is increased for each sample as long as $\alpha_c^i < \alpha_{ref}^{up}$. Similarly, as $\alpha_c^i \geq \alpha_{ref}^{up}$ we decrease the switch probability with a faster rate as

$$\beta_{c,c+1}^i(k) = \beta_{c,c+1}^i(k-1) - \hat{K}_r^{up}\left(\alpha_c^i(k) - \alpha_{ref}^{up}\right), \tag{3.5}$$

where $\hat{K}_r^{up} > K_r^{up}$

3.2.2.2 Channel Shrinking

Nodes switch to higher channels when their current channel gets congested. We also need a mechanism by which nodes may switch back to lower channels once the traffic is less busy. This will reduce the cost of cross-channel communication. Analogous to the case of advancing channels, this scheme has nodes inviting nodes from higher channels once the success rate, α, is above a given threshold, α_{ref}^{down}. The invitation probability for a node i at channel $c+1$ to switch down to the current channel c if $\alpha_c^i > \alpha_{ref}^{down}$ is given as

$$\beta_{c,c-1}^i(k) = \beta_{c,c-1}^i(k-1) + K_r^{down}\left(\alpha_c^i(k) - \alpha_{ref}^{down}\right) \tag{3.6}$$

As before we decrease probability with faster rate when we have $\alpha_c^i \leq \alpha_{ref}^{down}$ as

$$\beta_{c,c-1}^i(k) = \beta_{c,c-1}^i(k-1) - \hat{K}_r^{down}\left(\alpha_c^i(k) - \alpha_{ref}^{down}\right), \tag{3.7}$$

The key element in both channel expansion and shrinking is to accurately set the controller gain K^{up} and K^{down}, which determine how aggressively or conservatively switching occurs. (Higher K implies a higher switching probability or more aggressive switching.)

3.2.2.3 Choosing the Controller Gains

The integral controller increases the switch probability by a small fraction at a time until enough nodes have switched to improve the quality of the current channel. However, depending on the topology of the network, there may be a substantial delay before the effect of a channel switch of one node has propagated to others on the old channel. It is well-known from basic feedback control theory that delay decreases the stability of a feedback control system, since it takes longer for the effects of control actions to become measurable. As a result, there is an interesting trade-off to consider when choosing the switching probability.

If it is too big, switching is aggressive and nodes may oscillate among channels, moving back and forth excessively and causing overhead. On the other hand, if it is too small, it will take longer for the network to re-configure to new channels upon load changes. This trade-off is captured by the controller gain parameters K_r^{up} and K_r^{down}.

In the following, we will provide an insight into how to choose the gain parameters such that we avoid excessive channel fluctuations in the presence of delay. The analysis will be based on computing the worst-case switching frequency at network delay, d (measured in controller samples). The main reason for the delay comes from propagating the domino effect of switching from a sink back through intermediate nodes to the source. During this time, nodes close to the source still experience the same measured poor performance α as before the switch. The main effect of the delay is thus increased possibility that more nodes than necessary will switch to the new channel not knowing that someone else has already switched and that soon performance will consequently improve.

We will compute K_r^{up} and K_r^{down} such that the fraction of nodes that may switch during the propagation delay, d, is below a given threshold, γ. For channel expansion, the worst-case situation occurs if $\alpha_c^i = 0$, in which case the increase in the switch probability of Equation (3.4) from sample to sample is equal to $K_r^{up} \cdot \alpha_{ref}^{up}$. During the first sample, an average fraction $K_r^{up} \cdot \alpha_{ref}^{up}$ of the nodes will leave and $(1 - K_r^{up} \cdot \alpha_{ref}^{up})$ will remain at the channel. In the second sample, the switch probability will increase to $2 \cdot K_r^{up} \cdot \alpha_{ref}^{up}$ and the fraction of the original nodes that leave in this sample is equal to $(1 - K_r^{up} \cdot \alpha_{ref}^{up}) \cdot 2 \cdot K_r^{up} \cdot \alpha_{ref}^{up}$. The fraction of nodes, $\Gamma(d, K_r^{up}, \alpha_{ref}^{up})$, that switch channels during a time interval of d samples can, thus, be computed as

$$\Gamma(d, K_r^{up}, \alpha_{ref}^{up}) = \sum_{m=1}^{d} \left(m \cdot K_r^{up} \cdot \alpha_{ref}^{up} \cdot \prod_{j=1}^{m-1} \left(1 - j \cdot K_r^{up} \cdot \alpha_{ref}^{up}\right) \right) \tag{3.8}$$

For given values of α_{ref}^{up}, d, and γ (which all can be assumed to be available off-line), we

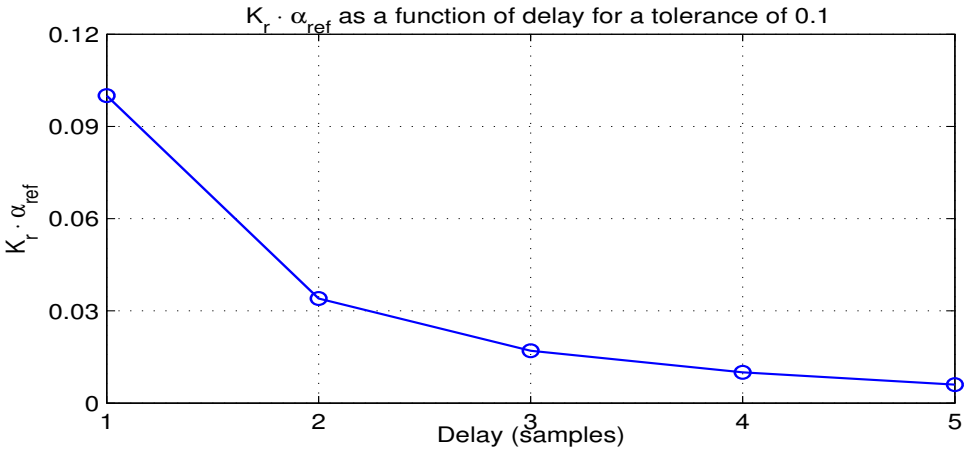


Figure 3.1: $K_r^{up} \cdot \alpha_{ref}^{up}$ as a function of the network delay for a tolerance of $\gamma = 0.1$.

may compute K_r^{up} from the relation $\Gamma(d, K_r^{up}, \alpha_{ref}^{up}) = \gamma$. As an example, Figure 3.1 shows $K_r^{up} \cdot \alpha_{ref}^{up}$ as a function of delay for the case $\gamma = 0.1$.

The same analysis applies for the case of channel shrinking, with the exception that the worst-case switch probability is given by $K_r^{down}(1 - \alpha_{ref}^{down})$.

With this analysis, we have provided a more intuitive design parameter than choosing the controller gains K_r^{up} and K_r^{down}. Specifying γ can be interpreted as choosing the worst-case fraction of nodes moving to a new channel. To prevent sustained oscillations, this fraction has to be less than 1. Smaller fractions have a larger stability margin (in a control-theoretic sense) but fractions that are too small cause a sluggish system response to load changes.

3.2.2.4 Channel Overflow

Once a home channel gets overcrowded, nodes switch to the upper channel. When the bandwidth of the available channels is sufficient, nodes in the network will be distributed from channel F_0 and up to the number of channels needed to accommodate the traffic. However, in the worst-case, there is still a chance that N channels are not sufficient to avoid network overload. Channel overflow also happens when a closely connected set of nodes (e.g.

a sink and its followers) does not fit into one channel.

We propose a scheme to choose the threshold α_{ref}^{up} for channels, which solves the channel overflow problem. The idea of the scheme is that the higher a channel is, the lower its threshold should be; and the threshold of the highest channel F_N should be zero. This make nodes become more conservative in switching every time they go up one channel. Nodes stop considering switching channels once they get to the highest boundary channel. Based on that, the threshold α_{ref}^{up} is chosen as follows

$$\alpha_{ref,c}^{up} = \begin{cases} \alpha_{ref,0}^{up} - c \cdot \epsilon & \text{if } c < N \\ 0 & \text{if } c = N \end{cases} \tag{3.9}$$

with $\alpha_{ref,0}^{up}$ is the threshold at channel F_0 (where everynode starts to function at) and $\alpha_{ref,c}^{up}$ is the threshold at channel F_c, and ϵ is a chosen constant.

3.3 Protocol Design

In this section we will describe the design of the multi-channel MAC protocol and how it extends the existing components of TinyOS-2.x.

3.3.1 Component Structure

The protocol is packaged in a component which exposes the same interface found in the TinyOS-2.x Packet Protocol. This will allow new applications developed for TinyOS-2.x to use the new MAC without any porting effort. Furthermore, since the multi-channel MAC protocol works on top of a single-channel MAC, we also decouple the services provided by the basic MAC from the multi-channel MAC so that it is independent of the platform-specific implementation. Figure 3.2 shows how the interfaces among layers are split to facilitate seamless integration with both virtual platforms provided by TOSSIM and an actual MAC provided for the CC2420 ChipCon radio in the MAC of the MicaZ motes. The white boxes are interfaces and the solid boxes are implemented. Other platforms can be integrated similarly by adding a thin layer on top of their MACs and provide the interface used by the multi-channel MAC.

3.3.2 Algorithm Design

There are important design decisions which need to be taken to ensure that the algorithm will be simple and efficient enough for WSNs applications. Following we describe these decisions.

Time-triggered Activity

The MAC protocol is designed to work in a time-triggered manner. In other words, rate control is achieved explicitly using an interval timer as opposed to implicitly by receipt of

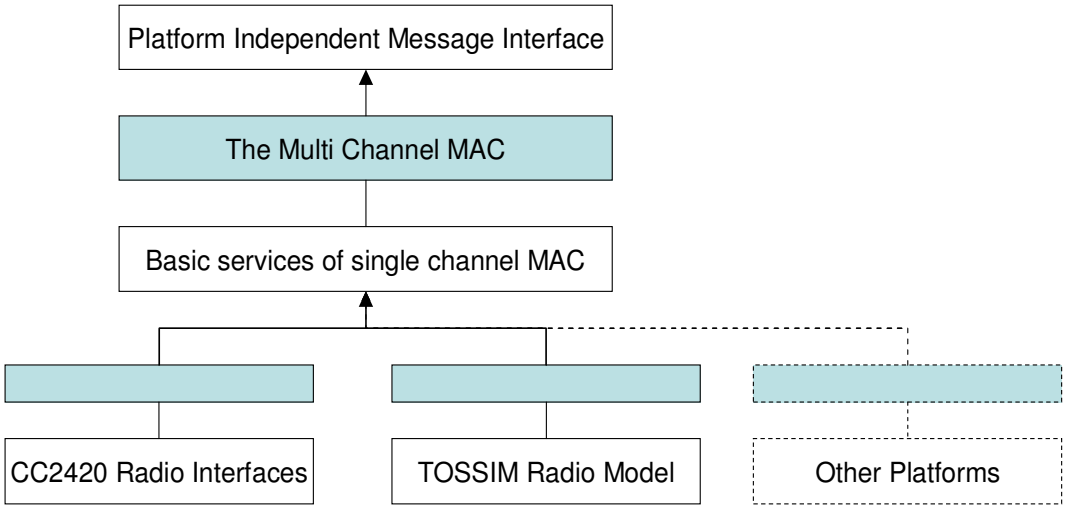


Figure 3.2: Component structure of the multi-channel MAC

send done notifications. Most message types are queued and served periodically (except a few types of messages as will be explained later). Special messages used in the protocol have places reserved in the network message queue so that data messages do not occupy the whole queue and prevent protocol messages from being sent.

Channel Status Updates

Nodes periodically broadcast their perceived home channel conditions. The information broadcasted out by a node i is a pair $< s_i, f_i >$ where s_i is the number of times the MAC layer succeeds in accessing the channel and f_i is the number of failed attempts. This pair together with the current home channel of node i will be put on the same message called a Channel Update Message, and is enqueued to the same queue as with normal messages.

Nodes collect channel update messages and use that information to estimate the channel acquisition probability as described earlier. From channel update messages, nodes also are able to know the up-to-date home channel of their neighbors.

Neighbors' Home Channel Maintenance

When a node wants to send a message to another node, it needs to switch to the home channel of the receiver before transmitting. Hence, a node needs to know the home channel of its neighbors who it communicates with. When nodes first join the network, they assume that the home channels of other nodes are the same as their own, which is F_0. When the home channel information is out of date, communication fails and the node initiates a search for neighbors on all channels as will be described later.

In the following, we will describe the different message types used by the MAC protocol, how these messages are queued, and the management of neighbor tables. We then give the functional description of the algorithm.

3.3.2.1 Message Types

We begin by describing the message types used by our protocol for future reference. When a node first joins the network, it broadcasts a HELLO message at the home channel to inform its neighborhood that it has joined the channel. When a node needs to send a message to a neighbor but does not know its home channel, it sends out WHERE IS messages. CHANNEL UPDATE messages are sent out periodically by nodes and contain the pair $< t_i, s_i >$ of every node i. These messages are sent at the home channel of the sender. BYE message are sent out by nodes that decide to leave their current channel (because of channel expansion or channel shrinking). When the home channel is underloaded (as described in Section 3.2) the node sends out INVITATION messages to the above channel to invite nodes to join its home channel. The last type of messages is DATA which constitutes any messages passed to the MAC by the upper layer via the component interface. The upper layer will be notified whenever a DATA message is sent successfully, or whenever the delivery failed for several transmission attempts.

Message Type	Included Information
WHERE IS	Home channel of the sender, Id of the requested node
WHERE IS RESPONSE	Home channel of the sender
HELLO	Old home channel
BYE	New home channel
CHANNEL UPDATE	$< t_i, s_i >$
INVITATION	$\beta_{c+1,c}$
DATA	Data

Table 3.1: Summary of the seven types of special protocol messages.

Table 3.1 gives a summary of the special network messages that are used by the MAC protocol.

The protocol traffic overhead happens in three cases: (i) updating channel status to nodes' neighbors, (ii) notifying neighbor about channel switching, and (iii) finding neighbors' home channels. In these overhead messages, the first type happens the most. In our implementation, channel update messages are sent periodically each one second. The length of this message is only 5 *bytes*. So, the overhead traffic is only around 5 *bytes/second/node*.

3.3.2.2 Message Queuing

Apart from WHERE IS, HELLO, and BYE messages, all types of messages – including WHERE IS RESPONSE, CHANNEL UPDATE, INVITATION, and DATA - are queued before being sent out. The MAC protocol periodically pops messages out of the queue and sends them to the corresponding destination. If the MAC fails to send a message, it will put the message back at the end of the queue. If the number of retry attempts exceeds a threshold, it will discard the message and notify the upper layer if the message is a data message.

Since the number of messages generated by the MAC (*protocol messages*) is small, and they play an important role in the behavior of the protocol, it is desirable for the queue to favor these messages over DATA messages from the upper layer. The queue implements

this by not allowing DATA messages to fill up the whole queue. The remaining spaces are reserved for protocol messages. By doing this, the protocol messages will rarely be discarded because of queue overflow, even in heavy traffic conditions[2].

3.3.2.3 Neighbor Table Management

Neighbor tables are required to maintain information about neighboring nodes that a node communicates with. The neighbor table is designed as a simple hash table in which keys are the neighbor IDs. Since the number of entries is finite, new entries will replace the entry which is least recently used when the table is full.

3.3.2.4 Functional Description

The overall algorithm is captured by the state machine shown in Figure 3.3[3] . After initialization, each node goes to an idle state from which it executes different actions depending on messages it receives and the expiration of timers.

After receiving a WHERE IS RESPONSE or WHERE IS message, the neighbor list is updated. In the latter case, a WHERE IS RESPONSE message is also popped to the top of the message queue before the node returns to the idle state.

A decision to switch the home channel is made each time the change home channel timer fires. In case the node decides to switch channels it executes the steps shown in the *Change Home Channel* subsystem of Figure 3.4.

The actions performed by the node are, first, to send out a BYE message at its current home channel, then switch to the new home channel, and finally send out a HELLO message on the new channel. After switching, the node returns to the idle state.

[2]The fact the messages are queued before transmitted should work with most of the application. There are chances that this may affect special applications as Deluge [16]. However, further study need to be done to conclude this

[3]Circles represent states, rounded boxes are processes, solid lines are conditional transitions, dotted lines are unconditional transitions, and diamonds are condition checks. Filled boxes represent composite processes

The second timer used in the implementation is for sending messages. As this timer fires, the first message is popped from the message queue. If the queue is non-empty, the next action is to determine if the home channel of the destination is known.

If the destination channel is unknown, the message is pushed back to the message queue if the number of transmission attempts does not exceed the maximum. Thereafter, WHERE IS messages are sent out on gradually increasing channels as described by the subsystem in Figure 3.5. The messages are sent starting at channel F_0 followed by all higher channels up to F_N until an acknowledgment is received. After the WHERE IS have been sent, the node returns to the idle state.

If the destination channel is known, the message is sent. The sending of a message is described in more detail in the subsystem of Figure 3.6. If the destination is on a different channel, the sender needs to switch channels before sending the message. After determining if an acknowledgment was received, the sender switches back to its home channel.

If the transmission was successful, the neighbor table is updated and the node returns to its idle state. If the transmission failed, the message is pushed back to the message queue if the number of transmissions is below the threshold.

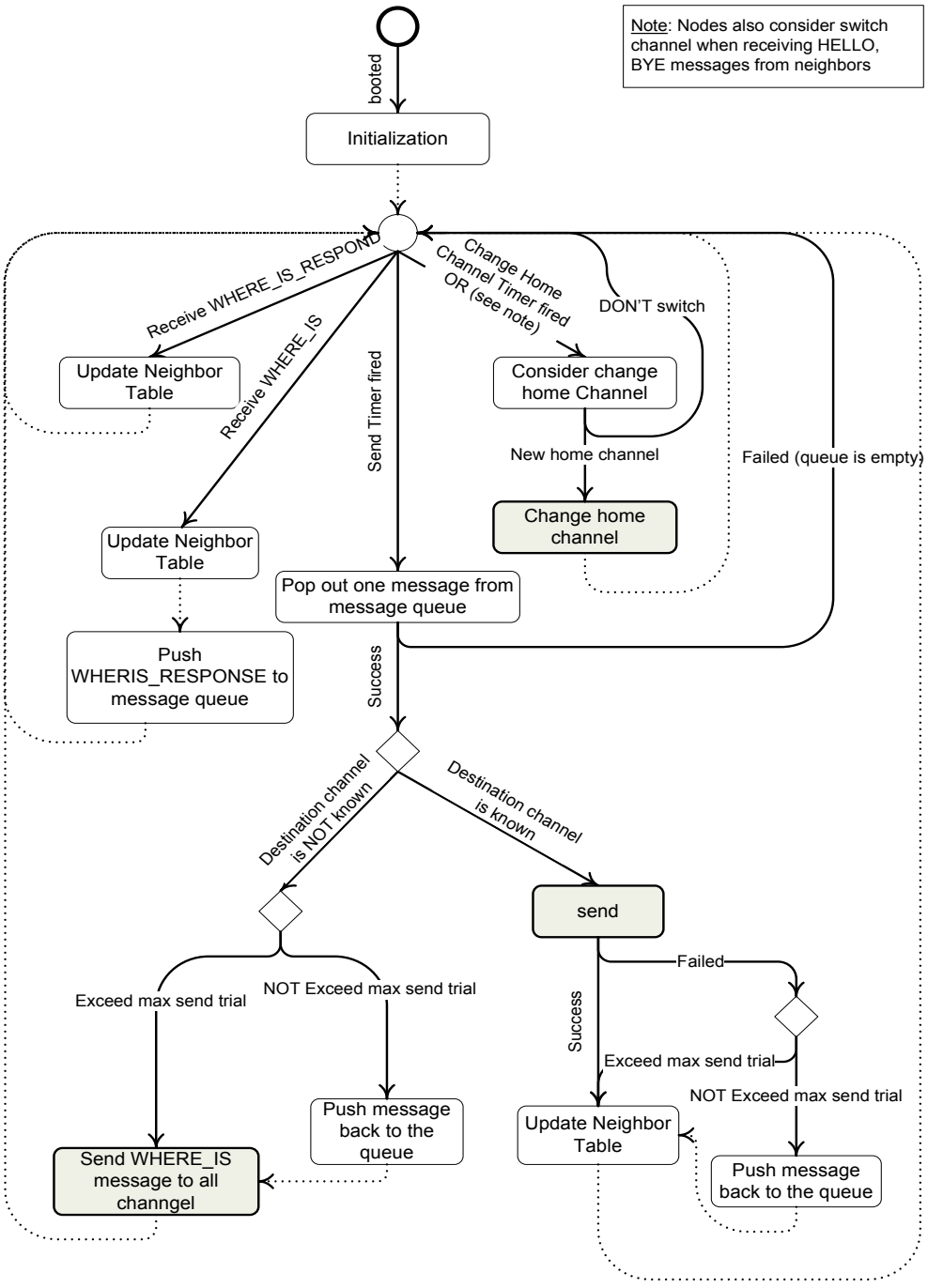


Figure 3.3: Overall state machine of the algorithm

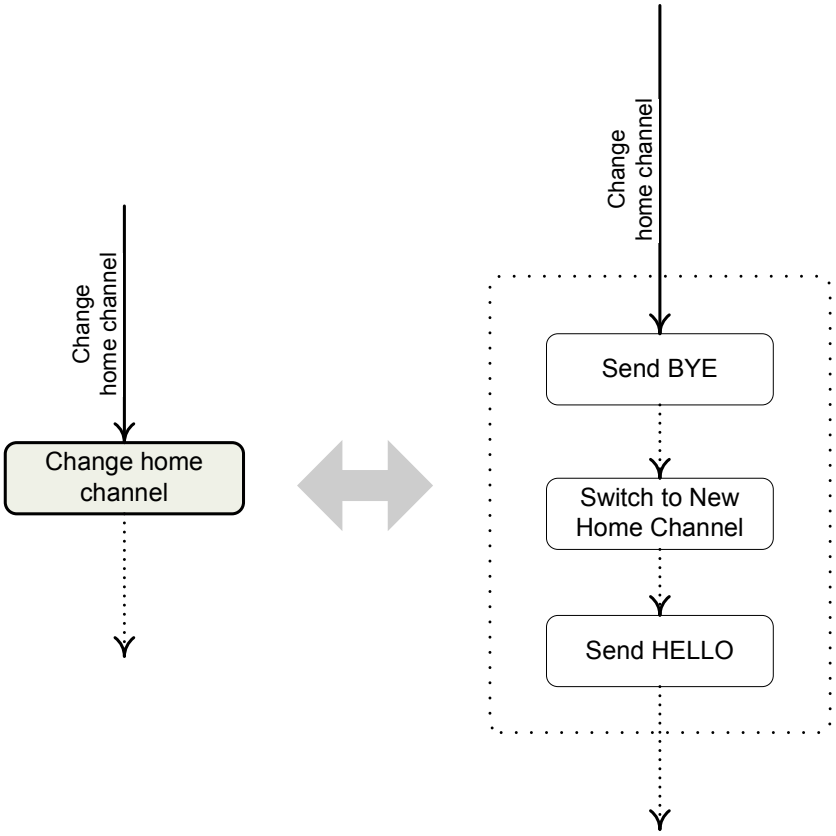


Figure 3.4: State machine subsystem for changing the home channel.

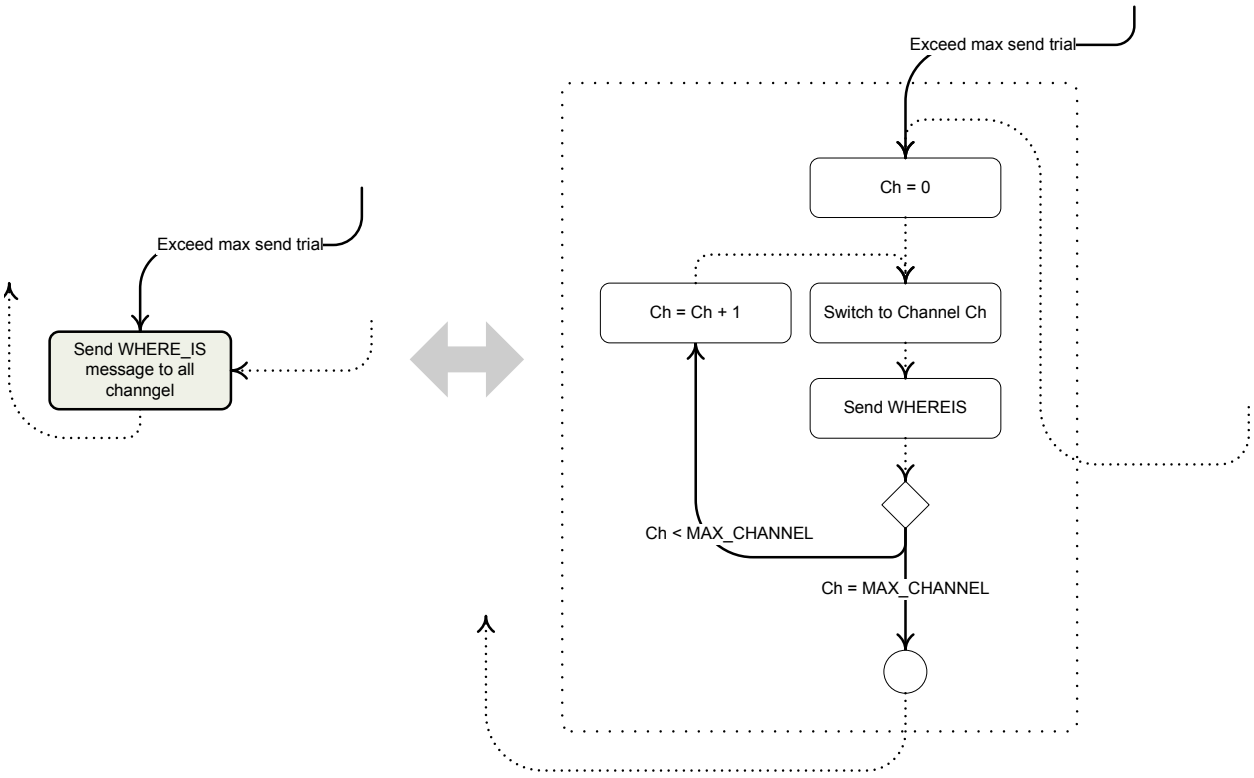


Figure 3.5: State machine subsystem for sending requests asking for the home channel of a neighbor.

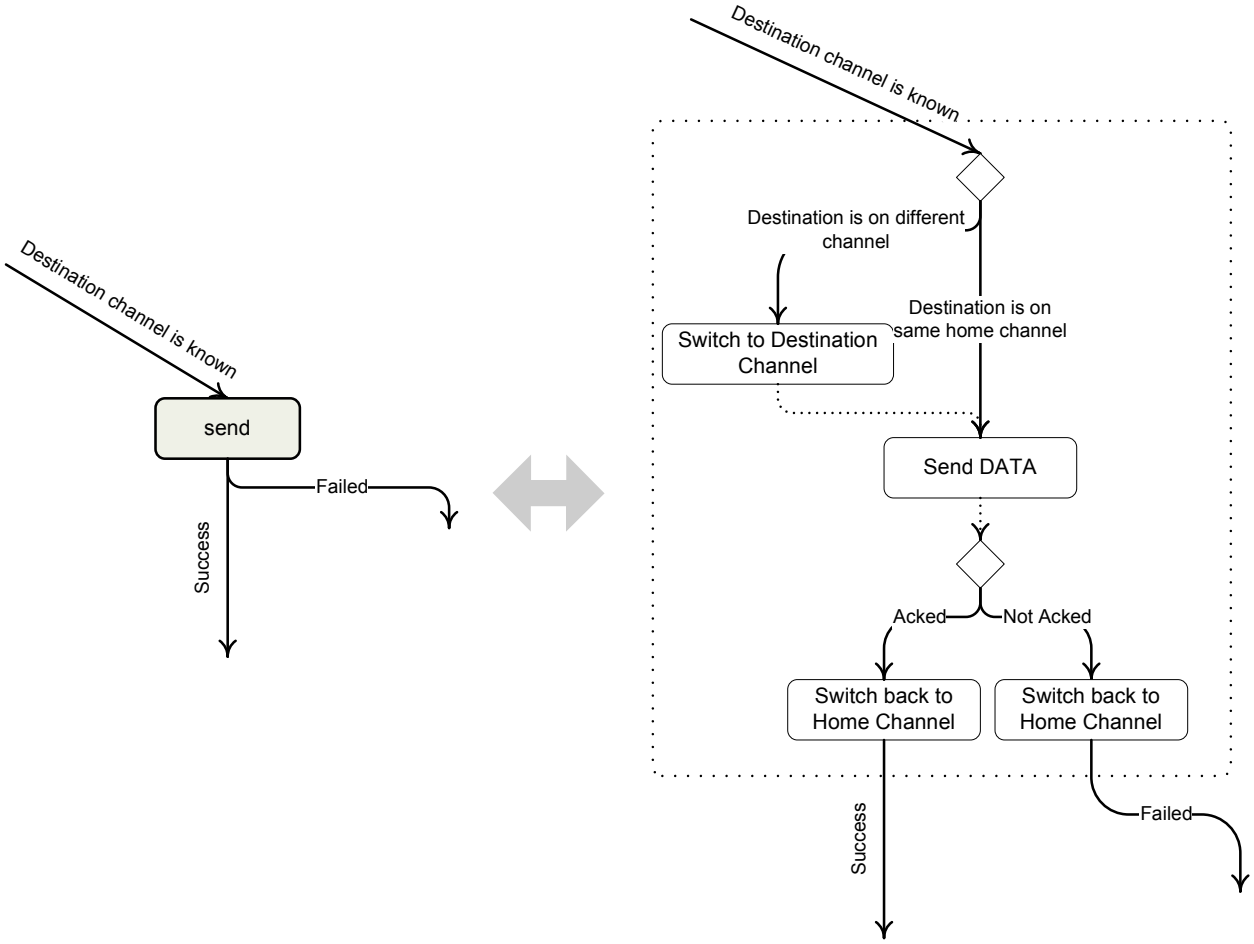


Figure 3.6: State machine subsystem for sending a message.

3.4 Implementation

In this section we give a brief overview of the implementation of the multi-channel MAC protocol on the MicaZ motes.

3.4.1 Code Structure and Footprint

Software components are created to conform to the design described in Section 3.3.1. The multi-channel MAC is implemented in the nesC programming language for MicaZ motes with TinyOS-2.x. The code is structured so that the platform-dependent parts are separated from the core mechanisms of the protocol. Therefore, in the core mechanism implementation, there is no distinction between TOSSIM and MicaZ. One code base is used for both platforms.

The compiled code for the multi-channel MAC is 9544 bytes in ROM and 761 bytes in RAM. In future work, we will optimize the footprint. In the rest of this section, we will briefly present technical issues related to enabling multi-channel communication on MicaZ motes as well as in TOSSIM [17].

3.4.2 Adopting Multi-Channel Communication Capabilities

We need the capability to communicate on different channels *dynamically at runtime*. TOSSIM does not support this directly. Hence, we had to modify TOSSIM to adopt this feature.

Making TOSSIM support dynamic channel switching requires introducing new types of events in the event queue as well as changing the implementation of the radio model. The newly introduced event for switching channels also takes the experimental channel switching time from real motes into account, which make the simulation model accurately reflect the physical constraint.

3.5 Evaluation

In this section, we present an evaluation of the multi-channel MAC both in testbed experiments with MicaZ motes and in simulation using TOSSIM. While the testbed results show the performance in a small-scale network setting, the evaluation in TOSSIM (with the same code base as is run on the MicaZ motes) enables testing at a larger scale.

The evaluation settings focus on collection and aggregation traffic patterns which are most popular in WSN. Admittedly, our protocol favors this case. This is not a coincidental choice. We believe that random point-to-point traffic patterns are less popular in WSN. Hence, they are neither targeted nor evaluated in this chapter and will generally result in poor performance of our protocol.

3.5.1 Experimental Testbed Evaluation

The following experimental evaluation will be run against simulations in TOSSIM for comparison and to validate the simulations with results from the real platform. The radio model is signal-strength-based. It follows previous literature [18], already supported in TOSSIM for TinyOS-2.x [19].

Figure 3.7: Setups used in the cross-channel communication evaluation.

3.5.1.1 Cross-channel Communication

We first evaluate a cross-channel communication scheme and compare the results from both testbed and TOSSIM. Figure 3.7 shows the experimental setups. In both setups, node 1 sends messages to node 2 and node 2 sends messages to node 3. In the first setup, all three

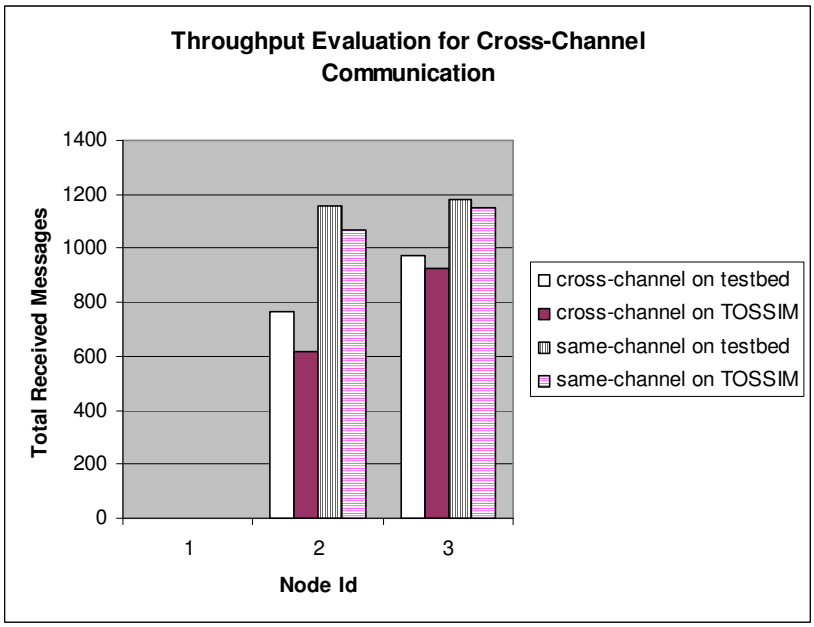


Figure 3.8: Throughput comparison in the cross-channel communication evaluation

nodes work on the same channel. In the second setup, the three nodes are assigned to three different channels. Hence, node 1 has to switch to the channel of node 2 to send and node 2 has to switch to the channel of node 3 to send. The deafness effect will happen in the traffic from node 1 to node 2.

Figure 3.8 shows the number of messages received at each node for the same-channel and cross-channel setups in both experiments and simulation. As can be seen, there is a good match between values from the testbed and values from the simulation. The most significant difference between simulations and the testbed is in the number of messages received in the cross-channel case. The lower throughput in the simulation is due to an over-estimation of the channel switching time. This makes our simulation model more pessimistic, while still valid for a comparison in the more realistic cases studied in Section 3.5.2.

Figure 3.9 shows another view in which we compare the ratio of throughput in the same-channel case and the cross-channel case for each node in the testbed and TOSSIM environments. The results for both message-generating rates (1 message/25 milliseconds and

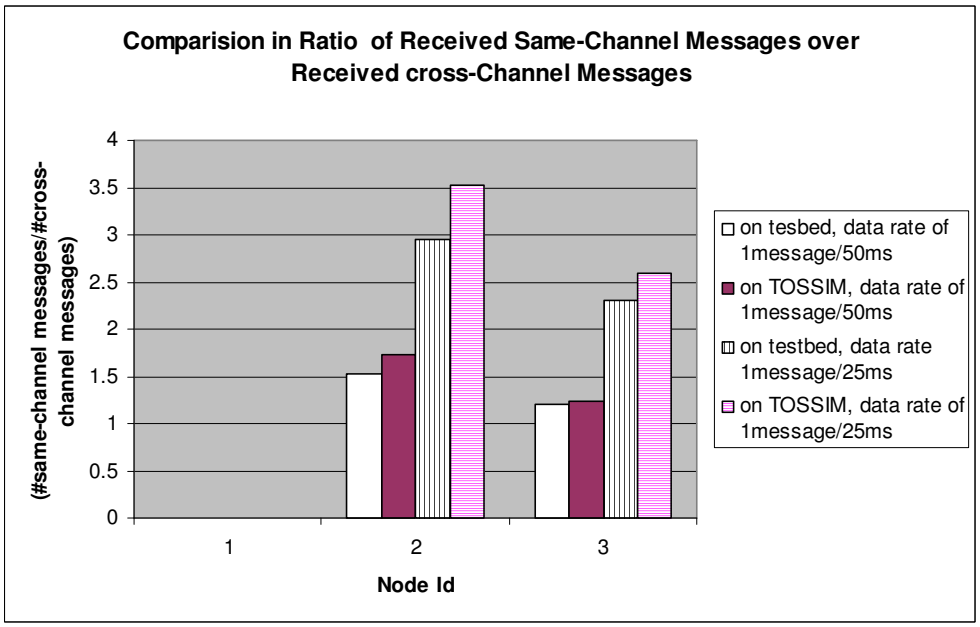


Figure 3.9: More comparision between TOSSIM and testbed results

1 message/50 milliseconds) again show a good match between the simulation and the real testbed.

3.5.1.2 Effect of Utilizing Multiple Channels

In this section, we evaluate how the multi-channel MAC improves throughput in a crowded network. A testbed with 16 nodes was used with the topology shown in Figure 3.10. The arrows show the traffic flows. The left figure shows the initial channel settings and the right figure shows the channel allocation after the network stabilizes. The experiment lasts for 10 minutes. Nodes reach the final channel allocation configuration on the right of Figure 3.10 in less than 3 minutes. The results in Figure 3.11 show that the case of using multiple channels on average outperforms the single channel case by about 30% in terms of throughput.

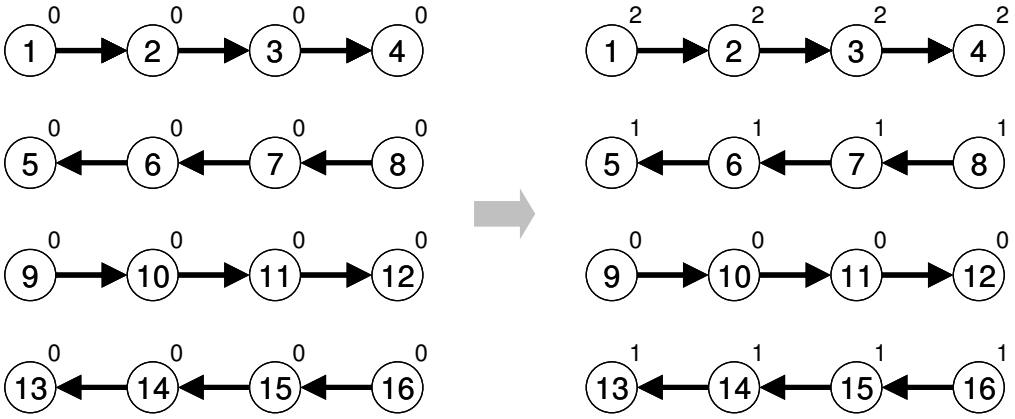


Figure 3.10: 16-node setup used for testbed experiments.

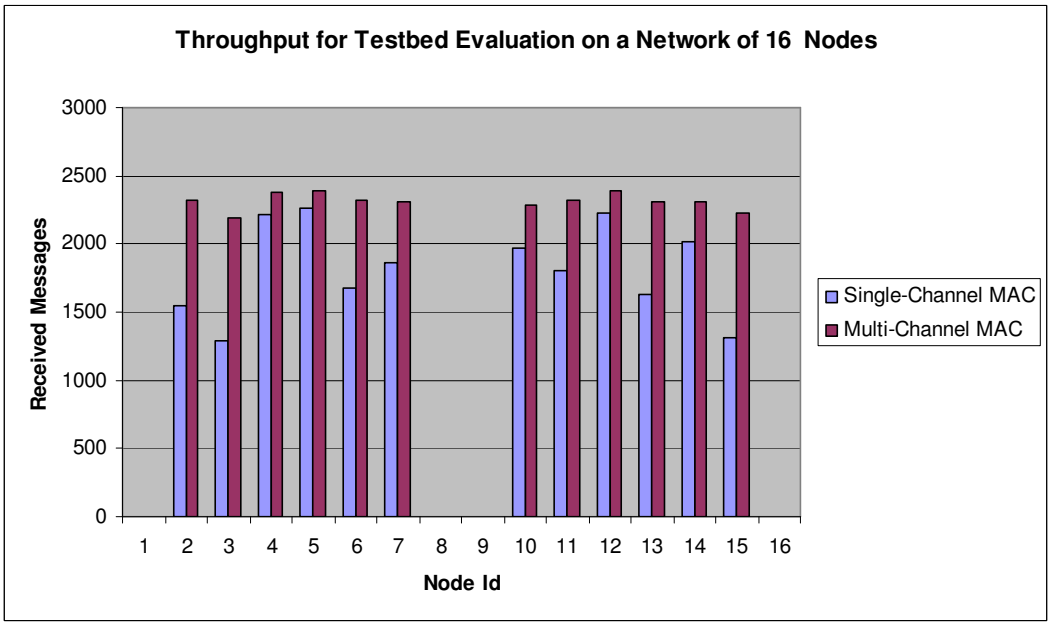


Figure 3.11: Testbed comparison between networks with a single channel MAC and with the multi-channel MAC for the 16-node setup.

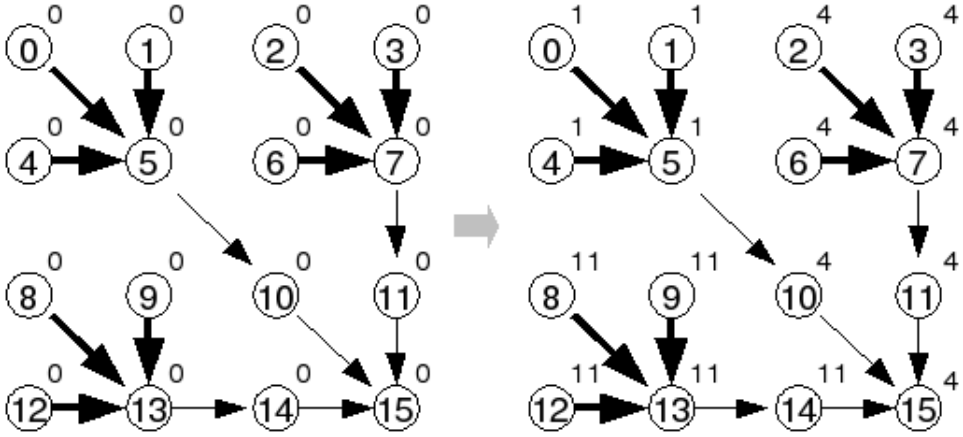


Figure 3.12: Network with tree lightly connected sub-networks

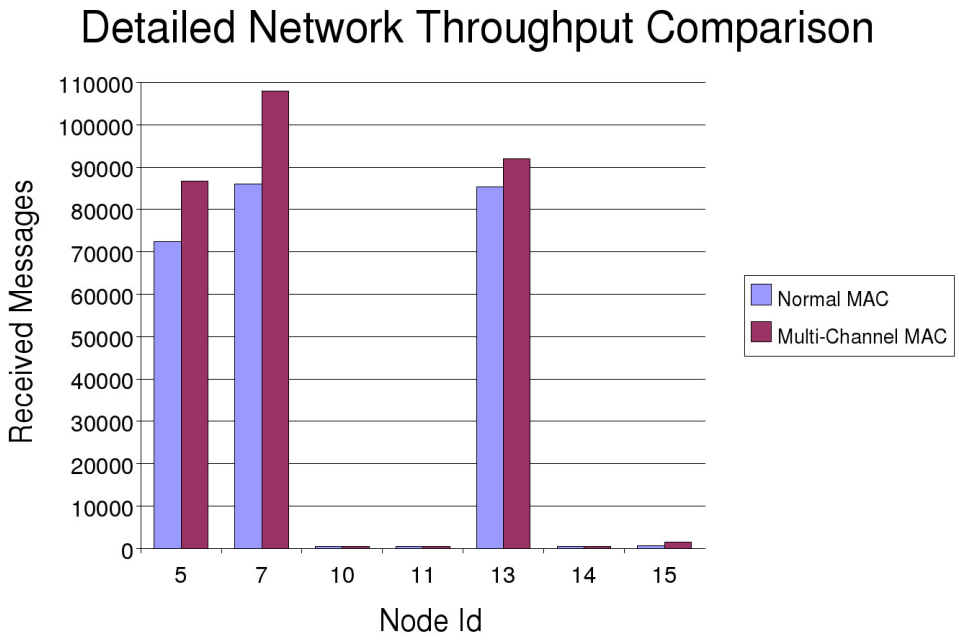


Figure 3.13: Testbed comparison between networks with a single channel MAC and with the multi-channel MAC for the 16-node setup.

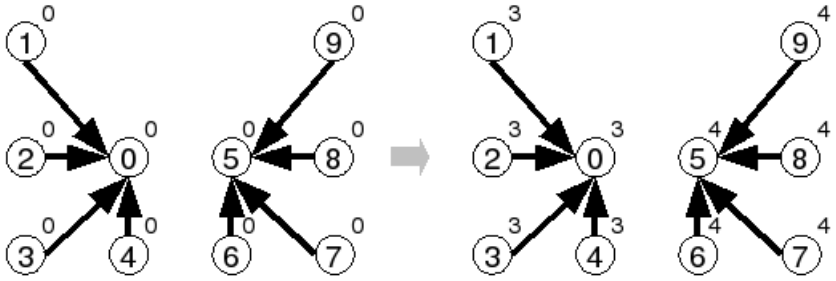


Figure 3.14: Networks with two separated sub-networks. The two sub-networks later are located on two different channels

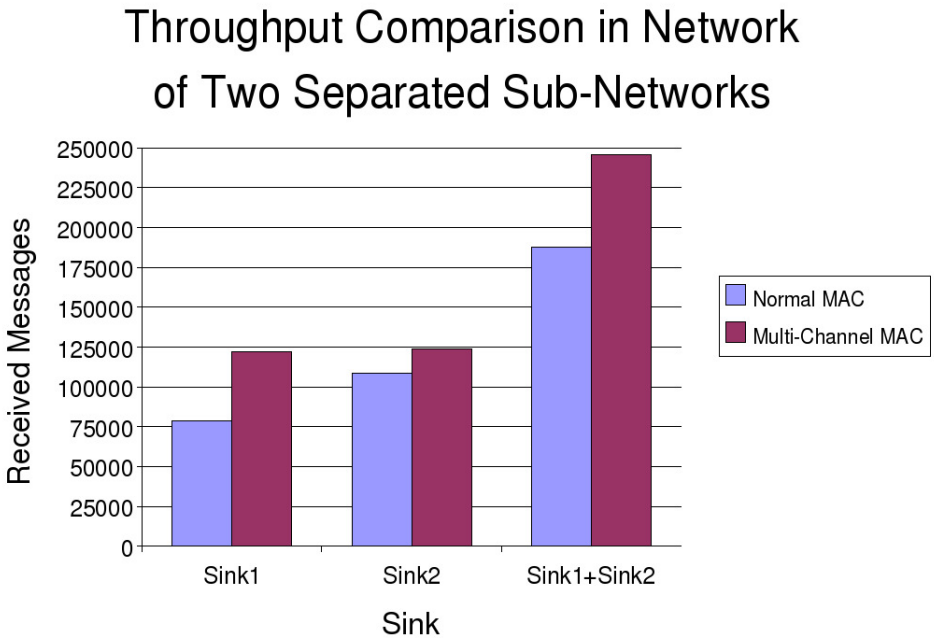


Figure 3.15: Testbed comparison in network with two separated sub-networks.

3.5.1.3 Network of Independent Sub-networks

We experiment with network traffic patterns shown in Figure 3.14. The network includes two separated sub-networks which form two different collection trees. The experiment was conducted in 10 minutes. After 3 minutes, the channel allocation was as shown on the right side of the figure. The two indepedent parts were located on two different channels. The throughput comparison is shown in Figure 3.15. We observe that the multi-channel MAC outperforms the single channel MAC by 31%. This experiment also shows the ability to avoid interference of the multi-channel MAC. Each sub-network can consider the other as a source of interference thus they end up at different channels.

3.5.1.4 Network of Lightly Connected Sub-networks

In this experiment, the network traffic is chosen as shown in Figure 3.12. This network is organized as an aggregation tree which includes three other sub-trees. Different data rates are also introduced in this evaluation (the thin arrows correspond to a data rate of 1 message / 1000 milliseconds, and the thick arrows have data rate of 1 message / 10 milliseconds). As shown in the throughput comparison in Figure 3.13, our MAC protocol out-performs the case of a single-channel MAC at all aggregation nodes. In particular, for node 16 the throughput improvement is roughly 25%.

3.5.2 Simulation and Scaling

The previous experimental evaluation in section 3.5.1.1 showed a good match between simulation and the real testbed. In this section, we scale the evaluation by simulation to a network of 36 nodes composed of two aggregation trees. The roots of the trees are placed next to each other so they interfere. The topology of the network is shown in Figure 3.16.

At the end of the simulation, the nodes in each aggregation tree end up at different

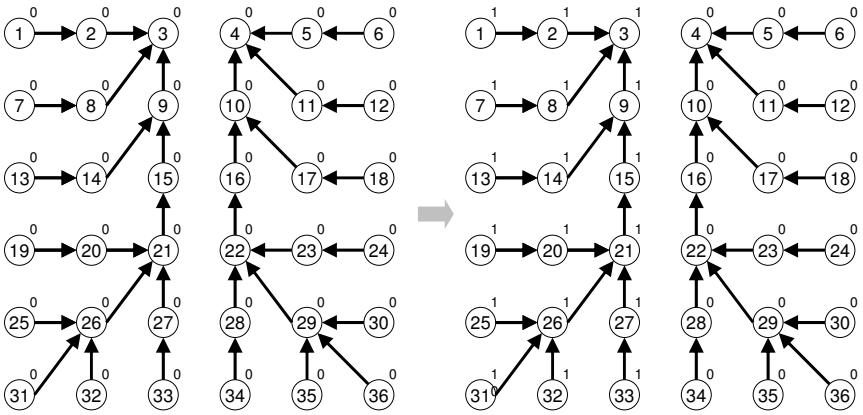


Figure 3.16: 36-node network with two aggregation trees placed close to each other.

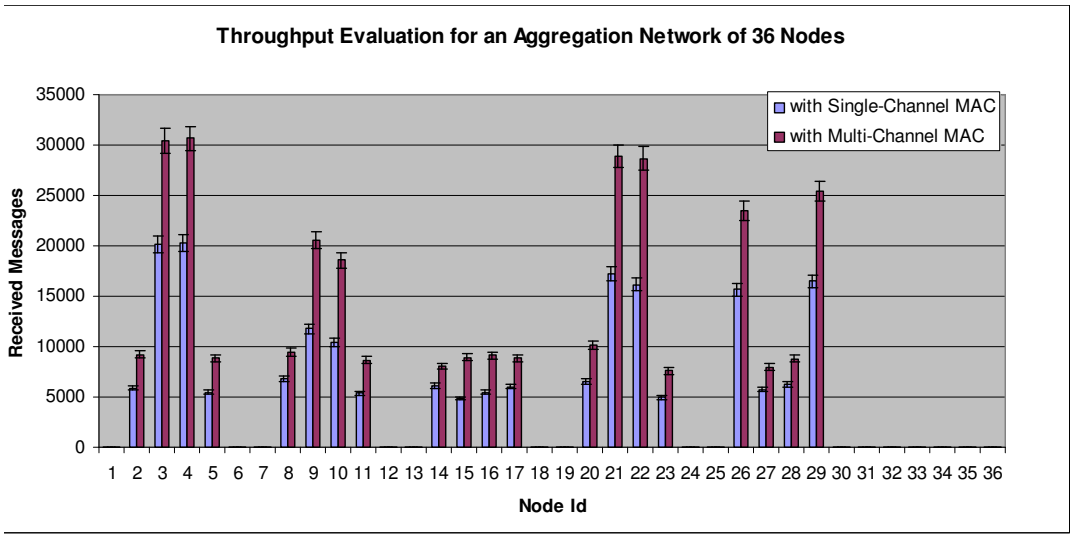


Figure 3.17: Throughput comparision for the 36-node aggregation network in simulation

channels, which helps the throughput improve considerably at each node. The throughput increases by roughly 50% at the aggregation points as shown in Figure 3.17.

3.6 Conclusions

This chapter presented a practical design, implementation, and evaluation of a multi-channel MAC protocol for wireless sensor networks (WSNs). The multi-channel MAC protocol constitutes the first real implementation that considers the hardware constraints (single half-duplex radio interfaces and non-trivial channel switching times) associated with commonly used WSN motes.

A distributed heuristic was proposed to partition nodes among channels in a way that keeps costly cross-channel communication to a minimum. Furthermore, a simple feedback control strategy was designed to oversee the partitioning process in a way that ensures stability and avoids congestion.

The MAC protocol is simple and light-weight and was evaluated on a proof-of-concept testbed with MicaZ motes. The evaluation showed that the multi-channel protocol was successful in avoiding network congestion and achieved performance improvements compared to the single-channel case.

In this work, we focused on improving network throughput. Message delay, message loss, and power management issues are left for furture studies.

PART II

Information and Social Network Layer

Once data is collected from sensors and other sources, the next step in the information pipeline is to clean up the data and eliminate unreliable items and sources. The data cleaning algorithms developed in this section are general and can be equally applied to time-series sensor data as well as to unstructured data from social sources. The wide adoption of social networking sites like Facebook, Twitter and Google+ makes social sources and microblogs an especially interesting new information input to our "data-to-decision" pipeline. Hence, we consider applications that must process data generated by (or with the help of) a large number of users and sensors. The generated data may have a vast differences in quality and reliability. We propose a scalable solution to distill such data into a managable amount of higher-quality information. This is done by assessing credibility of each reported information item and the credibility of the sources from which information was generated. While the techniques developed in this section are general, we shall focus more on unstructured data modalities, such as human text.

The part is organized as follows. In Chapter 4, we present concepts, theoretical foundation, and architecture of Apollo: a data distillation service for social sensing applications. In Chapter 5, we investigate in details the benefit of utilizing knowledge about sources to further prune lower quality information gathered from social networks. In Chapter 6, we present further approaches which make Apollo able to handle streams of incoming information in real-time and in a scalable manner.

CHAPTER 4

APOLLO: A DATA DISTILLATION SERVICE FOR SOCIAL SENSING

This chapter presents Apollo, a data distillation service for human-centric sensing applications. The goal of the service is to filter a deluge of incoming data by jointly estimating the credibility of sensing sources and observations made by them, then eliminating less credible observations. Social sensing applications, where participants volunteer data collection for the application server, are becoming an increasingly popular use case of mobile sensing platforms. A phone-based social sensing campaign may grow virally, generating a significant amount of data at the server that may be noisy, incorrect, or come from un-vetted sources. This motivates the design of server-side data distillation to perform data triage before acting on the data. Apollo provides such a general service in a largely application-independent manner. It is novel in that it can handle not only time-series sensor data, but also unstructured text or image data, which is a common format in human-centric sensing. Hence, Apollo can be readily integrated with a wide range of applications from use of text tweets (on Twitter) for reporting the progression of an event of interest, and use of imaging reports for locating community issues needing attention, to use of GPS traces for inferring traffic congestion on city streets.

4.1 Introduction

Apollo is a new service for social sensing applications, aimed at distilling large amounts of noisy social sensing data (received at the server) into smaller amounts of more trustable information. The service is motivated by the proliferation of mobile platforms equipped with common sensors (e.g., cameras, GPS, and acceleration tracking), Internet connectivity, and large-scale dissemination opportunities (e.g., via Twitter and Flickr), making such platforms ideal for social sensing applications. We refer by *social sensing* to those applications where humans act as the sensors, the sensor operators, or the sensor carriers. For example, humans may report (on Twitter) events in the environment that they monitor. They may document observation by operating camera-phones, or they may download cell-phone applications that transparently perform other sensor data collection and sharing.

A key challenge in social sensing applications is noisy data. Consider a smart-phone sensing application that grows virally in a community to collect data of mutual interest and share it with some central entity for processing and display. Participants in this application may not be *a priori* screened or vetted. They may vary widely in their degree of reliability. Moreover, their reliability may not be *a priori* known. The lack of explicit participant vetting and the corresponding data reliability and uncertainty problems pose difficulties in computing accurate conclusions from data. Extracting reliable observations from multitudes of possibly unreliable data sources, who may be unknown to the application in advance, emerges as a major challenge. We call this challenge *data distillation*. This study presents a data distillation solution for social sensing.

Since distillation functionality is needed in a broad category of social sensing applications, it is good to delegate it to a general service that many different applications can use. Apollo is an exercise in solving the distillation challenge in a generalized application-independent manner, such that the solution can be easily adopted by different applications without sig-

nificant effort, despite differences in each application's purpose, data formats, and exact cleaning needs.

Underlying the distillation challenge is the abstraction of *sources* (e.g., sensors and people) and *claims* (the observations they make). The goal is simply to select (i.e., "distill") from the pool of all claims only those that exceed a certain credibility threshold. If sources were equally reliable, providing a general data distillation solution would be trivial. One could simply count the number of sources who make a specific claim and assign a credibility value to that claim that grows with the number of sources. Hence, claims reported by more sources would be considered more believable and the distillation engine would prune other uncorroborated claims. This scheme is known as *voting*. Unfortunately, sources are not equally reliable, so voting does not always work well. Moreover, source credibility is not *a priori* known. Instead, Apollo utilizes a scheme where the credibility of both sources and claims is assessed jointly, allowing for more informed selection of believable claims.

The idea of joint estimation of credibility of sources and claims comes from recent fact-finding literature in the machine-learning community [20, 21]. Inspired by that literature, which focuses on ranking of sources and claims, we developed a scheme for computing the exact probability that a source is correct, based on a group of general simplifying assumptions [22]. Apollo uses this result to perform data distillation.

Apollo is not meant to outperform clever application-specific data cleaning schemes. There is little that can beat large amounts of careful application-specific knowledge. Instead, Apollo offers simplicity and generality. Rather than developing their own specialized data cleaning solutions, an application developer can simply use Apollo to do a good enough job. Apollo can therefore help reduce the cost barrier to developing new social sensing applications from noisy data. In scenarios where specialized application-specific data cleaning solutions are already available, Apollo may offer a scalability advantage. By implementing a first "quick and dirty" filter, it can eliminate from the data pool large parts that are less credible before

passing the remaining claims for more specialized analysis that is more time-consuming.

In this study, we made following contributions on addressing the above challenges of building a distributed real-time social information distillation engine:

1) We designed a general architecture and built a working system for the service. We evaluate our system in four representative scenarios of increasing complexity, that cover the range of roles humans play in data collection.

2) We developed a distillation analytics foundation based on fact-finding algorithms.

3) We investigated the benefit of source selection in social sensing applications.

4) We proposed two extensions of the system so that it can handle a large stream of real-time social information. In the fist extension, we extended the capability of Apollo so that it can work in a real-time manner. Credibility of sources is continuously updated over time. We aslo designed a user interface so that users can track events in different temporal granularity as well as different level of detail from the social data stream. In the second extension, a distributed version of the engine was designed to allows the system to handle stream of data with significantly greater size.

4.2 The Apollo Analytic Engine

Information networks are a key abstraction in data mining literature used to uncover facts from a large number of relations between unreliable observations [23]. The power of information network analysis lies in its ability to extract useful conclusions even when the degree of reliability of the input data or observations is not known in advance. For example, given a set of claims from a multitude of sources, one can rank both the claimed information pieces (let us call them *assertions*) and their sources by credibility, given no *a priori* knowledge of the truthfulness of the individual assertions and sources. Alternatively, given only data on who publishes in which conferences one can rank both the authors and the conferences by authority in the field.

This approach presents a new analytic framework that enables, for the first time, the calculation of correct probabilities of conclusions resulting from information network analysis. Such probabilities constitute a measure of quality of information (QoI). Our analysis relies on a Bayesian interpretation of the basic inference mechanism used for fact-finding in information network literature.

In the simplest version of fact-finding from information networks, nodes represent entities such as sources and assertions. Edges denote their relations (e.g., who claimed what). Each category of nodes is then iteratively ranked. Assertions are given a ranking that is proportional to the number of their sources, each source weighted by its credibility. Sources are then given a ranking that is proportional to the number of the assertions they made, each weighted by its credibility. This iterative ranking process continues until it converges. Information network analysis is good at such ranking. While the algorithms compute an intuitive "credibility score", as we demonstrate in this study, they do not actually compute the real *probability* that a particular conclusion is true. For example, given that some source is ranked 17th by credibility, it is not clear what that means in terms of probability that

the source says the truth. Our study addresses this problem, providing a general analytic foundation for quantifying the probability of correctness in fact-finding literature. We show that the probabilities computed using our analysis are significantly more accurate than prior work.

The fact-finding techniques addressed in this approach are particularly useful in environments where a large number of sources are used whose reliability is not *a priori* known (as opposed collecting information from a small number of well-characterized sources). Such situations are common when, for instance, crowd-sourcing is used to obtain information, or when information is to be gleaned from informal sources such as Twitter messages. We focus on networks of sources and assertions. The Bayesian interpretation derived in this study allows us to accurately quantify the probability that a source is truthful or that an assertion is true in the absence of detailed prior knowledge. Note that, while only source/assertion networks are considered, the analysis allows us to represent a much broader category of information networks. For example, in the author/conference network, one can interpret the act of publishing in a conference as an implicit assertion that the conference is good. The credibility of the assertion depends on the authority of the author. Hence, the network fits the source/assertion model.

This approach is intended to be a first step towards a new category of information network analysis. Being the first step, we focus on laying the foundations, such that extensions of this work can easily adapt the analysis to more complex information network models. Hence, we start with a very simple model in this approach, and leaving extensions to the next approach using expectation maximization.

Technical Details

We are specifically interested in the network model used for deriving credibility of facts and sources. We call the iterative ranking algorithm used for analyzing source/assertion

information networks, a *fact-finder*. The algorithm ranks a list of assertions and a list of sources by credibility. In the following subsections, we first review the basic algorithm, then propose its Bayesian interpretation that allows quantifying the actual probability that a source is truthful or that an assertion is true.

The Basic Fact-finder Let there be s sources, $S_1, ..., S_s$ who collectively assert c different pieces of information, $C_1, ..., C_c$. We call each such piece of information an assertion. We represent all sources and assertions by a network, where these sources and assertions are nodes, and where a claim, $C_{i,j}$ (denoting that a source S_i makes assertion C_j) is represented by a link between the corresponding source and assertion nodes. We assume that a claim can either be true or false. An example is "John Smith is CEO of Company X" or "Building Y is on Fire". We further define $Cred(S_i)$ as the credibility of source S_i, and $Cred(C_j)$ as the credibility of assertion C_j.

Algebraically, we define the $c \times 1$ vector, $\overline{C}_{cred}$, to be the assertion credibility vector $[Cred(C_1)...Cred(C_c)]^T$ and the $s \times 1$ vector, $\overline{S}_{cred}$, to be the source credibility vector $[Cred(S_1)...Cred(S_s)]^T$. We also define the $c \times s$ array CS such that element $CS(j,i) = 1$ if source S_i makes claim C_j, and is zero otherwise.

Now let us define $\overline{C}^{est}_{cred}$ as a vector of *estimated* assertion credibility, defined as $(1/\alpha)[CS]\overline{S}_{cred}$. One can pose the basic fact-finding problem as one of finding a least squares estimator (that minimizes the sum of squares of errors in source credibility estimates) for the following system:

$$\overline{C}^{est}_{cred} = \frac{1}{\alpha}[CS]\overline{S}_{cred} \tag{4.1}$$

$$\overline{S}_{cred} = \frac{1}{\beta}[CS]^T\overline{C}^{est}_{cred} + \overline{e} \tag{4.2}$$

where the notation X^T denotes the transpose of matrix X. It can further be shown that

the condition for it to minimize the error is that α and β be chosen such that their product is an Eigenvalue of $[CS]^T[CS]$. The algorithm produces the credibility values $Cred(S_i)$ and $Cred(C_j)$ for every source S_i and for every assertion C_j. These values are used for ranking. The question is, does the solution have an interpretation that allows quantifying the actual probability that a given source is truthful or that a given assertion is true? The question is answered in the next section.

A Bayesian Interpretation Let S_i^t denote the proposition that "Source S_i speaks the truth". Let C_j^t denote the proposition that "Assertion C_j is true". Also, let S_i^f and C_j^f denote the negation of the above propositions, respectively. Our objective, in this section, is to estimate the probabilities of these propositions. We further define S_iC_j to mean "Source S_i made assertion C_j".

It is useful to define $Claims_i$ as the set of all claims made by source S_i, and $Sources_j$ as the set of all sources who claimed assertion C_j. In the subsections below, we derive the posterior probability that an assertion is true, followed by the derivation of the posterior probability that a source is truthful.

Assertion Credibility Consider some assertion C_j, claimed by a set of sources $Sources_j$. Let i_k be the kth source in $Sources_j$, and let $|Sources_j| = K_j$. (For notational simplicity, we shall occasionally omit the subscript j from K_j in the discussion below, where no ambiguity arises.) According to Bayes theorem:

$$P(C_j^t|S_{i_1}C_j, S_{i_2}C_j, ..., S_{i_K}C_j) = \frac{P(S_{i_1}C_j, S_{i_2}C_j, ..., S_{i_K}C_j|C_j^t)}{P(S_{i_1}C_j, S_{i_2}C_j, ..., S_{i_K}C_j)} P(C_j^t) \tag{4.3}$$

The above equation makes the implicit assumption that the probability that a source makes any given assertion is sufficiently low that no appreciable change in posterior probability can

be derived from the lack of a claim (i.e., lack of an edge between a source and an assertion). Hence, only existence of claims is taken into account. Assuming further that sources are conditionally independent (i.e., given an assertion, the odds that two sources claim it are independent), Equation (4.3) is rewritten as:

$$P(C_j^t|S_{i_1}C_j, S_{i_2}C_j, ..., S_{i_K}C_j) = \frac{P(S_{i_1}C_j|C_j^t)...P(S_{i_K}C_j|C_j^t)}{P(S_{i_1}C_j, S_{i_2}C_j, ..., S_{i_K}C_j)}P(C_j^t) \tag{4.4}$$

Let us further assume that the change in posterior probability we get from any single source or claim is small. This is typical when using evidence collected from many individually unreliable sources. Hence:

$$\frac{P(S_{i_k}C_j|C_j^t)}{P(S_{i_k}C_j)} = 1 + \delta_{i_kj}^t \tag{4.5}$$

where $|\delta_{i_kj}^t| << 1$. Similarly:

$$\frac{P(S_{i_k}C_j|C_j^f)}{P(S_{i_k}C_j)} = 1 + \delta_{i_kj}^f \tag{4.6}$$

where $|\delta_{i_kj}^f| << 1$. Under the above assumptions, we prove in Appendix A that the denominator of the right hand side in Equation (4.4) can be rewritten as follows:

$$P(S_{i_1}C_j, S_{i_2}C_j, ..., S_{i_K}C_j) \approx \prod_{k=1}^{K_j} P(S_{i_k}C_j) \tag{4.7}$$

Please see Appendix A for a proof of Equation (4.7). Note that, the proof does *not* rely on an independence assumption of the marginals, $P(S_{i_k}C_j)$. Those marginals are, in fact, not independent. The proof merely shows that, under the assumptions stated in Equation (4.5)

and Equation (4.6), the above approximation holds true. Substituting in Equation (4.4):

$$P(C_j^t|S_{i_1}C_j, S_{i_2}C_j, ..., S_{i_K}C_j) = \frac{P(S_{i_1}C_j|C_j^t)...P(S_{i_K}C_j|C_j^t)}{P(S_{i_1}C_j)...P(S_{i_K}C_j)}P(C_j^t) \tag{4.8}$$

which can be rewritten as:

$$\begin{aligned} P(C_j^t|S_{i_1}C_j, S_{i_2}C_j, ..., S_{i_K}C_j) &= \frac{P(S_{i_1}C_j|C_j^t)}{P(S_{i_1}C_j)} \\ &\times \quad ... \\ &\times \frac{P(S_{i_K}C_j|C_j^t)}{P(S_{i_K}C_j)} \\ &\times P(C_j^t) \end{aligned} \tag{4.9}$$

Substituting from Equation (4.5):

$$\begin{aligned} P(C_j^t|S_{i_1}C_j, S_{i_2}C_j, ..., S_{i_K}C_j) &= P(C_j^t)\prod_{k=1}^{K_j}(1+\delta_{i_kj}^t) \\ &= P(C_j^t)(1+\sum_{k=1}^{K_j}\delta_{i_kj}^t) \end{aligned} \tag{4.10}$$

The last line above is true because higher products of $\delta_{i_kj}^t$ can be neglected, since we assumed $|\delta_{i_kj}^t| << 1$. The above equation can be re-written as:

$$\frac{P(C_j^t|S_{i_1}C_j, S_{i_2}C_j, ..., S_{i_K}C_j) - P(C_j^t)}{P(C_j^t)} = \sum_{k=1}^{K_j}\delta_{i_kj}^t \tag{4.11}$$

where, from Equation (4.5):

$$\delta^t_{i_k j} = \frac{P(S_{i_k}C_j|C^t_j) - P(S_{i_k}C_j)}{P(S_{i_k}C_j)} \tag{4.12}$$

Source Credibility Next, consider some source S_i, who makes the set of claims $Claims_i$. Let j_k be the kth claim in $Claims_i$, and let $|Claims_i| = L_i$. (For notational simplicity, we shall occasionally omit the subscript i from L_i in the discussion below, where no ambiguity arises.) According to Bayes theorem:

$$\begin{aligned} P(S^t_i|S_iC_{j_1}, S_iC_{j_2}, ..., S_iC_{j_L}) \quad &= \\ \frac{P(S_iC_{j_1}, S_iC_{j_2}, ..., S_iC_{j_L}|S^t_i)}{P(S_iC_{j_1}, S_iC_{j_2}, ..., S_iC_{j_L})} &P(S^t_i) \end{aligned} \tag{4.13}$$

As before, assuming conditional independence:

$$\begin{aligned} P(S^t_i|S_iC_{j_1}, S_iC_{j_2}, ..., S_iC_{j_L}) \quad &= \\ \frac{P(S_iC_{j_1}|S^t_i)...P(S_iC_{j_L}|S^t_i)}{P(S_iC_{j_1}, S_iC_{j_2}, ..., S_iC_{j_L})} &P(S^t_i) \end{aligned} \tag{4.14}$$

Once more we invoke the assumption that the change in posterior probability caused from any single claim is very small, we get:

$$\frac{P(S_iC_{j_k}|S^t_i)}{P(S_iC_{j_k})} = 1 + \eta^t_{ij_k} \tag{4.15}$$

where $|\eta^t_{ij_k}| << 1$. Similarly to the proof in Appendix A, this leads to:

$$\begin{aligned} P(S^t_i|S_iC_{j_1}, S_iC_{j_2}, ..., S_iC_{j_L}) &= \frac{P(S_iC_{j_1}|S^t_i)}{P(S_iC_{j_1})} \\ &\times \ ... \\ &\times \frac{P(S_iC_{j_L}|S^t_i)}{P(S_iC_{j_L})} \\ &\times P(S^t_i) \end{aligned} \tag{4.16}$$

We can then re-write Equation (4.16) as follows:

$$\begin{aligned} P(S^t_i|S_iC_{j_1}, S_iC_{j_2}, ..., S_iC_{j_L}) &= P(S^t_i)\prod_{k=1}^{L_i}(1+\eta^t_{ij_k}) \\ &= P(S^t_i)(1+\sum_{k=1}^{L_i}\eta^t_{ij_k}) \end{aligned} \tag{4.17}$$

The above equation can be further re-written as:

$$\frac{P(S^t_i|S_iC_{j_1}, S_iC_{j_2}, ..., S_iC_{j_L}) - P(S^t_i)}{P(S^t_i)} = \sum_{k=1}^{L_i}\eta^t_{ij_k} \tag{4.18}$$

where, from Equation (4.15):

$$\eta^t_{ij_k} = \frac{P(S_iC_{j_k}|S^t_i) - P(S_iC_{j_k})}{P(S_iC_{j_k})} \tag{4.19}$$

The Iterative Algorithm In the sections above, we derived the expressions of posterior probability that an assertion is true or that a source is truthful. These expressions were derived in terms of $\delta^t_{i_kj}$ and $\eta^t_{ij_k}$. It remains to show how these quantities are related. Let us first consider the terms in Equation (4.12) that defines δ^t_{ikj}. The first is $P(S_iC_j|C^t_j)$, the probability that S_i claims assertion C_j, given that C_j is true. (For notational simplicity, we

shall use subscripts i and j to denote the source and the assertion.) We have:

$$P(S_iC_j|C_j^t) = \frac{P(S_iC_j, C_j^t)}{P(C_j^t)} \tag{4.20}$$

where:

$$\begin{aligned} P(S_iC_j, C_j^t) &= P(S_i\ speaks) \\ &\quad P(S_i\ claims\ C_j|S_i\ speaks) \\ &\quad P(C_j^t|S_i\ speaks, S_i\ claims\ C_j) \end{aligned} \tag{4.21}$$

In other words, the joint probability that link S_iC_j exists and C_j is true is the product of the probability that S_i speaks, denoted $P(S_i\ speaks)$, the probability that it claims C_j given that it speaks, denoted $P(S_i\ claims\ C_j|S_i\ speaks)$, and the probability that the assertion is true, given that it is claimed by S_i, denoted $P(C_j^t|S_i\ speaks, S_i\ claims\ C_j)$. Here, $P(S_i\ speaks)$ depends on the rate at which S_i makes assertions. Some sources may be more prolific than others. $P(S_i\ claims\ C_j|S_i\ speaks)$ is simply $1/c$, where c is the total number of assertions. Finally, $P(C_j^t|S_i\ speaks, S_i\ claims\ C_j)$ is the probability that S_i is truthful. Since we do not know ground truth, we estimate that probability by the best information we have, which is $P(S_i^t|S_iC_{j_1}, S_iC_{j_2}, ..., S_iC_{j_L})$. Thus:

$$P(S_iC_j, C_j^t) = \frac{P(S_i\ speaks)P(S_i^t|S_iC_{j_1}, S_iC_{j_2}, ..., S_iC_{j_L})}{c} \tag{4.22}$$

Substituting in Equation (4.20) from Equation (4.22) and noting that $P(C_j^t)$ is simply the ratio of true assertions, c_{true} to the total assertions, c, we get:

$$P(S_iC_j|C_j^t) = \frac{P(S_i\ speaks)P(S_i^t|S_iC_{j_1}, S_iC_{j_2}, ..., S_iC_{j_L})}{c_{true}} \tag{4.23}$$

Similarly,

$$P(S_iC_j) = \frac{P(S_i \ speaks)}{c} \tag{4.24}$$

Substituting from Equation (4.23) and Equation (4.24) into Equation (4.12) and re-arranging, we get:

$$\begin{aligned} \delta^t_{i_kj} &= \frac{P(S_{i_k}C_j|C^t_j) - P(S_{i_k}C_j)}{P(S_{i_k}C_j)} \\ &= \frac{P(S^t_i|S_iC_{j_1}, S_iC_{j_2}, ..., S_iC_{j_L})}{c_{true}/c} - 1 \end{aligned} \tag{4.25}$$

If we take the fraction of all true assertions to the total number of assertions as the prior probability that a source is truthful, $P(S^t_i)$ (which is a reasonable initial guess in the absence of further evidence), then the above equation can be re-written as:

$$\delta^t_{i_kj} = \frac{P(S^t_i|S_iC_{j_1}, S_iC_{j_2}, ..., S_iC_{j_L})}{P(S^t_i)} - 1 \tag{4.26}$$

Substituting for $\delta^t_{i_kj}$ in Equation (4.11), we get:

$$\begin{aligned} \frac{P(C^t_j|S_{i_1}C_j, S_{i_2}C_j, ..., S_{i_K}C_j) - P(C^t_j)}{P(C^t_j)} \quad = \\ \sum_{i=1}^{K_j} \frac{P(S^t_i|S_iC_{j_1}, S_iC_{j_2}, ..., S_iC_{j_L}) - P(S^t_i)}{P(S^t_i)} \end{aligned} \tag{4.27}$$

We can similarly prove that:

$$\eta^t_{ij_k} = \frac{P(C^t_j|S_{i_1}C_j, S_{i_2}C_j, ..., S_{i_K}C_j)}{P(C^t_j)} - 1 \tag{4.28}$$

and:

$$\frac{P(S_i^t|S_iC_{j_1}, S_iC_{j_2}, ..., S_iC_{j_L}) - P(S_i^t)}{P(S_i^t)} = \sum_{j=1}^{L_i} \frac{P(C_j^t|S_{i_1}C_j, S_{i_2}C_j, ..., S_{i_K}C_j) - P(C_j^t)}{P(C_j^t)} \tag{4.29}$$

Comparing the above equations to the iterative formulation of the basic fact-finder, described in Section 4.2, we arrive at the sought interpretation of the credibility rank of sources $Rank(S_i)$ and credibility rank of assertions $Rank(C_j)$ in iterative fact-finding. Namely:

$$Rank(C_j) = \frac{P(C_j^t|S_{i_1}C_j, S_{i_2}C_j, ..., S_{i_K}C_j) - P(C_j^t)}{P(C_j^t)} \tag{4.30}$$

$$Rank(S_i) = \frac{P(S_i^t|S_iC_{j_1}, S_iC_{j_2}, ..., S_iC_{j_L}) - P(S_i^t)}{P(S_i^t)} \tag{4.31}$$

In other words, $Rank(C_j)$ is interpreted as the increase in the posterior probability that an assertion is true, normalized by the prior. Similarly, $Rank(S_i)$ is interpreted as the increase in the posterior probability that a source is truthful, normalized by the prior. Substituting from Equation (4.30) and Equation (4.31) into Equation (4.27) and Equation (4.29), we then get:

$$\begin{aligned} Rank(C_j) &= \sum_{k \in Sources_j} Rank(S_k) \\ Rank(S_i) &= \sum_{k \in Claims_i} Rank(C_k) \end{aligned} \tag{4.32}$$

Once the credibility ranks are computed such that they satisfy the above equations (and

any other problem constraints), Equation (4.30) and Equation (4.31), together with the assumption that prior probability that an assertion is true is initialized to $p_a^t = c_{true}/c$, give us one of the important result, Namely[1]:

$$P(C_j^t|network) = p_a^t(Rank(C_j) + 1) \tag{4.33}$$

We can similarly show that if p_s^t is the prior probability that a randomly chosen source tells the truth, then:

$$P(S_i^t|network) = p_s^t(Rank(S_i) + 1) \tag{4.34}$$

Hence, the above Bayesian analysis presents, for the first time, a basis for estimating the probability that *each individual source*, S_i, is truthful and that *each individual assertion*, C_j, is true. These two vectors are computed based on two scalar constants: p_a^t and p_s^t, which represent estimated statistical averages over all assertions and all sources, respectively.

Evaluation Results

In this section, we carry out experiments to verify the correctness and accuracy of the probability that a source is truthful or an assertion is true predicted from the Bayesian interpretation of fact-finding in information networks. We then compare our techniques to previous algorithms in fact-finder literature.

We built a simulator in Matlab 7.8.0 to simulate the source and assertion information network. To test our results, we generate a random number of sources and assertions, and partition these assertions into true and false ones. A random probability, P_i, is assigned to each source S_i representing the ground truth probability that the source speaks the truth. For each source S_i, we then generate L_i claims. Each claim has a probability P_i of being

[1]The equations above are ambiguous with respect to a scale factor. To handle the ambiguity we impose the constraint that probabilities cannot exceed one.

true and a probability $1 - P_i$ of being false. A true claim links the source to a randomly chosen true assertion (representing that the source made that assertion). A false claim links the source to a randomly chosen false assertion. This generates an information network.

We let P_i be uniformly distributed between 0.5 and 1 in our experiments[2]. We then find an assignment of credibility values that satisfies Equation (4.32) for the topology of the generated information network. Finally, we compute the estimated probability that an assertion is true or a source is truthful from the resulting credibility values of assertions and sources based on Equation (4.33) and (4.34). Since we assumed that claims are either true or false, we view each assertion as "true" or "false" based on whether the probability that it is true is above or below 50%. Then the computed results are compared against the ground truth to report the prediction accuracy.

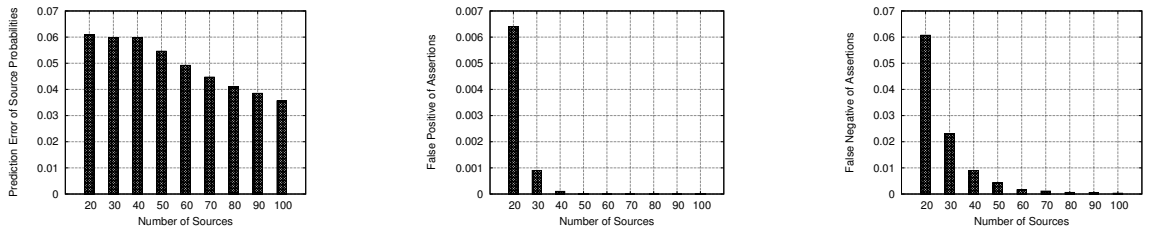
For sources, we simply compare the computed probability to the ground truth probability that they tell the truth. For assertions, we define two metrics to evaluate prediction accuracy: false positives and false negatives. The false positives are defined as the ratio of the number of false assertions that are classified as true over the total number of assertions that are classified as true. The false negatives are defined as the ratio of the number of true assertions that are classified as false over the total number of assertions that are classified as false. For each given source correctness probability (i.e., ground truth) distribution, we average the results over 100 network topologies (e.g., datasets over a time series). Reported results are averaged over 100 random source correctness probability distributions.

In the first experiment, we show the effect of the number of sources on prediction accuracy. We fix the number of true and false assertions at 1000 respectively. We set the average number of claims per source to 100. The number of sources is varied from 20 to 100. The prediction accuracy for both sources and assertions is shown in Figure 4.1. We note that both

[2]In principle, there is no incentive for a source to lie more than 50% of the time, since negating their statements would then give a more accurate truth

false positives and false negatives decrease as the number of sources grows. For more than 40 sources less than 1% of assertions are misclassified. The source correctness probability prediction exhibits a relatively small error (between 3% and 6%). The error first increases and then decreases as the number of sources increases. The reason is that there are two conflicting factors that affect the credibility prediction accuracy of sources: i) average number of assertions per source and ii) average number of sources per assertion. As the number of sources increases, the first factor decreases (reduce source credibility prediction accuracy) and the second factor increases (improve assertion and eventually source credibility prediction accuracy). When the number of sources is small, the change of the first factor is more significant than the second, thus its effect dominates. As the number of sources increases, the effect of the second factor overweights the first one and makes source correctness probability prediction error reduce.

Note that, the source correctness probability prediction is especially accurate (e.g., error is around 0.03) when the number of sources is relatively large. At the same time, both the false positives and false negatives in assertion classification are near zero under those conditions, illustrating that the approach has good scalability properties. Its usefulness increases for large networks.

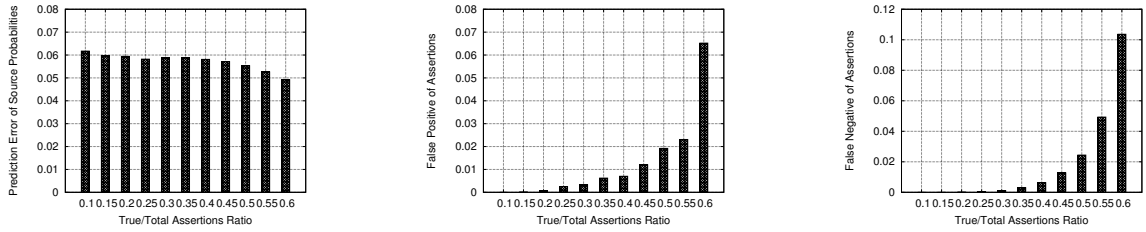


(a) Source Prediction Accuracy (b) Assertion Prediction False Positives (c) Assertion Prediction False Negatives

Figure 4.1: Prediction Accuracy vs Varying Number of Sources

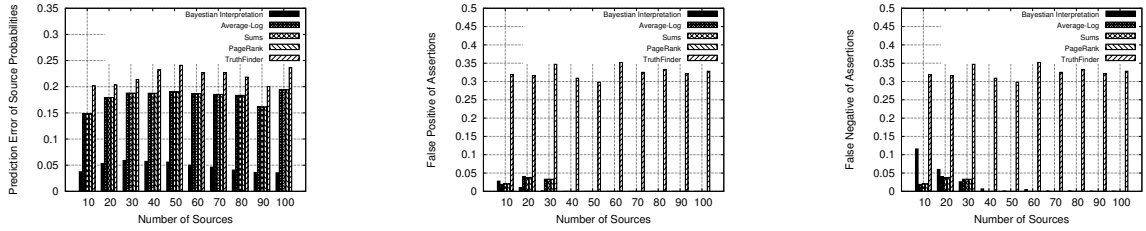
The next experiment shows the effect of changing the assertion mix on prediction accuracy. We vary the ratio of the number of true assertions to the total number of assertions in the

network. Assuming that there is usually only one variant of the truth, whereas rumors have more versions, one might expect the set of true assertions to be smaller than the set of false ones. Hence, we fix the total number of assertions to be 2000 and change the ratio of true to total assertions from 0.1 to 0.6. The number of sources in the network is set to 30. The prediction accuracy for both sources and assertions is shown in Figure 4.2. Observe that the source correctness probability prediction error decreases as the ratio of true assertions increases. This is intuitive: more independent true assertions can be used to improve credibility estimates of sources. Additionally, the false positives and false negatives increase because the true assertion set becomes less densely claimed and more true and false assertions are misclassified as each other as the number of true assertions grows.



(a) Source Prediction Accuracy (b) Assertion Prediction False Positives (c) Assertion Prediction False Negatives

Figure 4.2: Prediction Accuracy vs Varying True/Total Assertions



(a) Source Prediction Accuracy (b) Assertion Prediction False Positives (c) Assertion Prediction False Negatives

Figure 4.3: Prediction Accuracy Comparison with Baseline Fact-finders

Finally, we compared our proposed Bayesian interpretation scheme to four other fact-

finder schemes: Average-Log [24], Sums(Hubs and Authorities) [25], an adapted PageRank [26] where claims are bidirectional "links" between source and asserted "documents", and TruthFinder [27]. We selected these because, unlike other state-of-art fact-finders (e.g., 3-Estimates [20]), these do not require knowing what mutual exclusion, if any, exists among the assertions. In this experiment, the number of true and false assertions is 1000 respectively, the number of claims per source is 100, and the number of sources varies from 20 to 100. The fact-finder baselines treat multiple source-assertion network topologies as a whole data set. Using the initial assertion beliefs suggested by [24], we ran each baseline fact-finder for 20 iterations, and then selected the 1000 highest-belief assertions as those predicted to be correct. The estimated probability of each source making a true claim was thus calculated as the proportion of predicted-correct claims asserted relative to the total number of claims asserted by source.

The compared results are shown in Figure 4.3. Observe that the prediction error of source correctness probability by the Bayesian interpretation scheme is significantly lower than all baseline fact-finder schemes. The reason is that these schemes are primarily designed with ranking in mind. Hence, while the ranking of sources by credibility may be correct, they may suffer a bias in computing the exact probability of correctness. In contrast, the Bayesian analysis estimates the source correctness probability based on Equation (4.34) derived in this chapter, which is the contribution of this work. Note that, our technique does not offer an advantage in classifying assertions as true or false, although it is generally as good as the baselines. This is acceptable since the other techniques excel at ranking, which (together with a hint on the number of correct assertions) is sufficient to identify the correct ones. The results illustrate the advantages of Equation (4.34).

Limitation of the Bayesian Approach The contributions of Equation (4.34) should be interpretted with caution. There are several ways in which our evaluation is limited, as

described below:

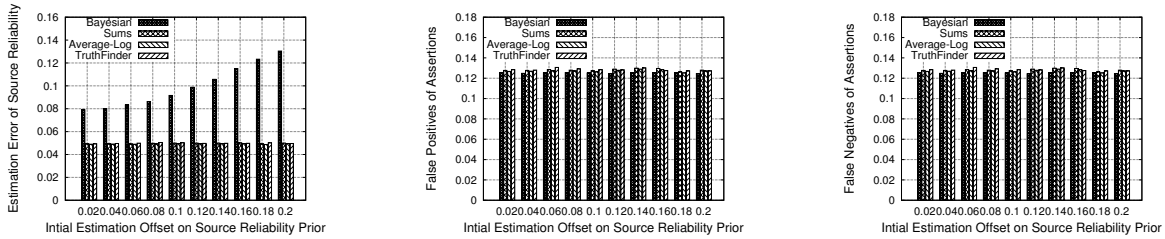
- First, the results are computed for a synthetic topology of the source-assertion network. In this topology, each source is connected to a number of assertions at random with a uniform distribution. In reality, some sources may be more likely to make certain claims, leading to skewed source-claim topologies that may change the results. This effect has not been investigated.

- Second, we assumed that all assertions are independent. Hence, knowing the truth value of one does not shed light on values of others. This is a very special case, as most claims are often related. For example, the assertion that it is "rainy" may also imply that it is "cloudy". Other baselines we compared against are much better at understanding and exploiting such relations. Hence, by assuming independence, we essentially created an advantage in favor of our scheme.

- Third, we have not exploited the power of natural language processing. For example, an assertion that there is a "fire" and that something is "burning" may corroborate each other. Hence, a practical implementation of our scheme is not straightforward, as it will need to understand such synonyms.

- Fourth, the approach is sensitive to prior assumptions. The following results show what happens when the source reliability prior, p_s^t (averaged over all sources), is not correctly estimated. In this case, it can be seen that the Bayesian method is less accurate than other fact-finders, even regarding on source reliability. Hence, the Bayesian interpretation approach is sensitive to the prior of average reliability of all sources in order to accurately predict the individual reliability of each source.

The simulation experiment was setup with 50 sources; with number of True and False assertions as 1000 each respectively. The number of average claims per source is 150. Correct



(a) Source Prediction Accuracy (b) Assertion Prediction False Positives (c) Assertion Prediction False Negatives

Figure 4.4: Effect of Error in Prior in Prediction Accuracy Comparison with Baseline Fact-finders

source reliability prior: 0.75; Initial estimation offset on the source reliability prior: $0.02-0.2$.

We observe that the Bayesian approach estimate less accurately than other fact-finders in source reliability and the difference becomes larger as the initial estimation offset on source reliability increases. The detail result is shown in figure 4.4. This is because Bayesian interpretation depends largely on the correct prior on source reliability for accurate estimation. At the same time, we observe that Bayesian achieve similar false positives/negatives as other fact-finders, this is because both Bayesian and other fact-finders depend on the correct prior on assertions to make decisions on assertion correctness, which is independent of initial source reliability prior offset.

Discussion

We just presented a Bayesian interpretation of the most basic fact-finding algorithm. The question was to understand why the algorithm is successful at ranking, and to use that understanding to translate the ranking into actual probabilities. Several simplifying assumptions were made that offer opportunities for future extensions.

No dependencies were assumed among different sources or different claims. In reality, sources could be influenced by other sources. Claims could fall into mutual exclusion sets, such as when one is the negation of the other. Taking such relations into account can further

improve quality of fact-finding. The change in posterior probabilities due to any single edge in the source-assertion network was assumed to be very small. In other words, we assumed that $|\delta^t_{i_kj}| << 1$ and $|\eta^t_{ij_k}| << 1$. It is interesting to extend the scheme to situations where a mix of reliable and unreliable sources is used. In this case, assertions from reliable sources can help improve the determination of credibility of other sources.

The probability that any one source makes any one assertion was assumed to be low. Hence, the lack of an edge between a source and an assertion did not offer useful information. There may be cases, however, when the absence of a link between a source and an assertion is important. For example, when a source is expected to bear on an issue, a source that "withholds the truth" exhibits absence of a link that needs to be accounted for.

Having the basic Bayesian interpretation in place, we can relax the above assumptions and accommodate the mentioned extensions in future work. While the iterative relations may become more cumbersome, we hope that the general outline of the proof can still be followed for these more complex scenarios.

4.3 Architecture of A Data Distillation Service for Social Sensing

Apollo aims to facilitate developing human-centric sensing applications by minimizing the effort application developers need to spend on cleaning unreliable data and by increasing the scalability of data cleaning. The analytics engine presented in the previous section does not provide all necessary components to interface with real-world applications. This leaves a number of questions in the design of social sensing systems:

- First, how to collect data in a general and application-independent fashion?
- Second, how to convert the collected data, which may consist of a range of data types, into an application-independent form suitable for distillation?

- Third, how to cluster various presentations referring to the same actual claims together so that the network of sources and claims can be constructed properly before being fed to the analytics engine?

In the rest of this section, we discuss our approach to addressing these questions. At a high-level, to offer data collection and distillation as a service, we create a system with two classes of components: application-independent and application-specific. The aim is to maximize the number and functionality of application-independent components, while minimizing and simplifying the application-specific ones. This has to be achieved in such a way that the quality of the results is not compromised. In particular, the quality of results should be comparable to (or at least not much worse than) that of custom-built solutions. In the next three sections, we briefly overview our approach for addressing the above three questions, respectively.

Data Collection

To address the first question (that of ensuring ease of data collection), the data collection mechanism needs to support a large system composed of human or sensory data sources that send human-centric observations that may include numerical values, text, or images. A general data collection back-end should be able to easily search and collect observations of interest independent of the data type. We find Twitter feeds to be a well-deployed infrastructure that supports such functionality. Using Twitter as the underlying engine for sharing sensing data allows for rapid integration with new applications. On the surface, Twitter offers a service for sharing text messages in real-time. However, sensors can also tweet their measurements. This is especially convenient if measurements are performed by human-carried sensors, such as those mounted on cell-phones, which constitute a large part of participatory and social sensing data sources. Finally, Twitter has also been used to share

pictorial data. Cell-phone applications such as TwitPic allow pictures taken by cell-phones to be uploaded to a TwitPic server, with links to them disseminated via Twitter. Apollo supports Twitter as a data collection tool. This makes it easier to integrate existing cell-phone-based sensory data collection applications that use Twitter without having to modify them to use new specialized protocols. Having said so, Apollo also supports a specialized data collection protocol that sends data to the Apollo server over a standard TCP/IP connection, should the application developer decide that the clients need to use it (by downloading the Apollo protocol client).

Data Conversion

To address the second question (data conversion to application-independent types), we represent the reported observations by a graph of sources and claims that we call the *source-claim network*. Sources simply refer to the IDs of devices that report the corresponding data. Claims can be thought of as abstract objects reported by the sources. In the source-claim network, a link between a source and a claim indicates that the source asserted that claim. The structure and semantics of the claim objects need not be known to Apollo. All that is needed is a measure of *distance* between claims. An application-specific part, called the *parser*, is responsible for converting observations into a unified format for claims that includes a claim ID and a pointer to a claim object. A distance function is also defined. It returns an application-specific notion of distance between claims. For example, if claims refer to sensory data, such as temperature values, the distance function is simply the difference in temperature. If claims refer to pieces of text, the distance function could be the Jaccard distance [28], a commonly used metric for deciding how similar two pieces of text are. Finally, if claims refer to images taken, a distance metric might be the color correlogram [29] or another visual similarity metric in vision literature. Note that, claims can also be multi-dimensional. For example, if temperature is sensed at different locations and different times

of day, one can think of these measurements as points in a space whose dimensions are sensor value (temperature), location, and time of day. A distance metric such as an (appropriately weighted) L2 norm can be defined between points in that space.

The source-claim network is a general representation of reported sensory data that enables cleaning. Other than the initial parser that converts the original data format into the source-claim representation, the only application-specific component that is needed for data distillation is the distance function. Once the source-claim graph is formed and distances are computed, the rest of the distillation task is application-independent.

Clustering

The first step in the application-independent distillation process is to perform clustering of claims. Clustering is an important step that significantly improves the scalability and quality of the process. In a real-world human-centric sensing application, sources will typically report slightly different observations, even when they measure the same variable or observe the same event. This results in a large number of (slightly different) individual claims and a large, poorly-connected, source-claim network, which has at least two negative consequences. First, it impairs scalability of the distillation algorithm (and increases its convergence time). Second, it lowers the quality of outputs because similar claims are treated separately and cannot get the credibility boost they would have enjoyed had they been considered together. Clustering of similar claims alleviates the above problems. It results in smaller well-connected source-claim networks in which joint estimation of source and claim correctness converges rapidly and the outcome becomes more accurate. The output of the clustering algorithm is a graph of individual sources and claim clusters (we call *consolidated claims*) that is input to the generic algorithm for joint estimation of source and claim correctness.

With the solutions for each of aforementioned questions provided, together with the core analytics engine, we have all essential components of a social data distillation service. Fig-

ure 4.5 illustrates the functional architecture of Apollo and the expectation maximization process.

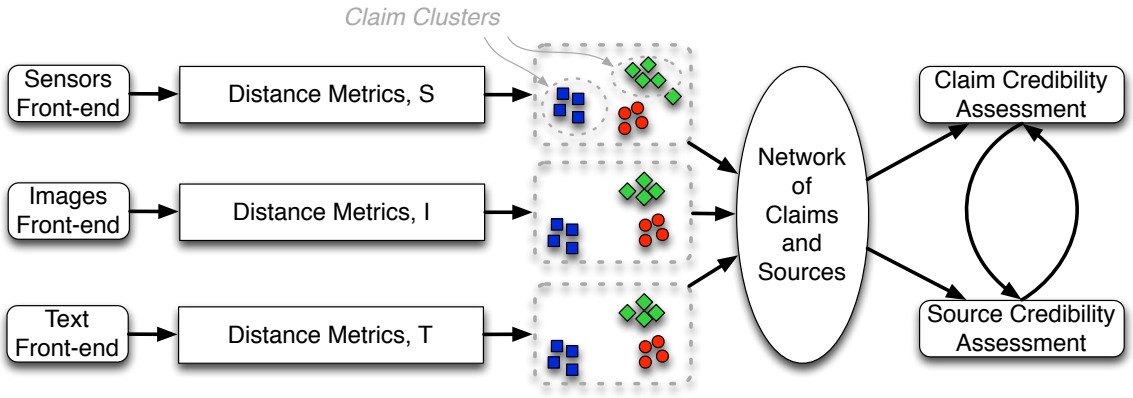


Figure 4.5: The architecture of Apollo.

4.3.1 Evaluation Results

We evaluate our system in four representative scenarios of increasing complexity, that cover the range of roles humans play in data collection. In the first, humans act as both sensors and sensor operators. Specifically, we consider geotagging applications, where locations of specific occurrences of interest are captured by users. Apollo automatically decides which reports are true and which are not. In the second, humans act as sensor operators. Users use cameras on their mobile devices to capture pictures and report their concern/attention. Apollo automacitally inspects and decides which pictures are relevant to the sensing task. In the third, humans act as sensor carriers. A set of participants simply allow GPS sensors in their possession to automatically report traffic speed data. Without direct human supervision, we show that the data reported is noisy enough that computing average traffic statistics from it is not always accurate. However, when data are clustered and ranked by Apollo, the quality of reported average statistics increases considerably. In the fourth and last scenario, we use Twitter data collected during uprisings of the "Arab Spring" (namely, the Cairo unrest

in the first two weeks of February 2011, when more than 1,000,000 tweets were collected). In this case, humans act as sensors, themselves. A significant number of raw text reports describe a much smaller number of events on Tahrir Square; some real and some rumored. We use Apollo to identify the distinct events from raw text and assign credibility values to them. We compare the results to media reports showing great correspondence with ground truth.

In all cases, we show that Apollo produces results that are more accurate than simple voting or averaging. The intuitive reason is that voting and averaging do not take source credibility into account. Apollo, jointly computes both source credibility and credibility of observations made by sources. Hence, it results in much more accurate estimates of probability of correctness of different claims.

We evaluate Apollo and show how different human-centric sensing applications with a variety of data types can be easily implemented using our data distillation service. For each application, we first describe the implementation of application-specific components, then we compare data cleaning with Apollo to other relevant baselines. Four social sensing applications are described in the order of complexity from the simplest to the most elaborate:

- *Geo-tagging:* In this "toy" example, we represent the general category of applications where participants search for occurrences of observations of interest and upload to a central server the location tags of such observations. Detecting invasive species in the "What's Invasive" campaign [30] is an instance of such applications.

- *PictureMe:* This application extends the above simple case by allowing users to report pictures reflecting concerns about a neighborhood or a local community. Apollo is used with a distance function that compares image content and location to identify those that should deserve more attention.

- *Speed Mapping:* In this application, individuals carry GPS devices (cell phones) and

share speed data to compute vehicular traffic conditions. They can be in or out of their cars. Various conditions such as faulty sensors or collecting data while not driving can contribute to inaccurate speed estimation. Apollo is used to distinguish between invalid and accurate driving speed data.

- *Human Sensors:* In this application, humans act as the sensors, simply tweeting about what they see. Apollo identifies what it believes to be the most credible tweets. We show that Apollo can identify important events as they occur and matches well events reported by the media.

In the rest of this section, we describe and evaluate each application separately. There are two observations to remember at this point. First, the purpose of this evaluation is to understand how well Apollo can distill data. Hence, we are concerned with knowing ground truth. For this reason (except in the raw Twitter data case), we run controlled experiments, where we know the reliable and unreliable participants upfront, and hence know ground truth by design. Second, it is not the purpose of this evaluation to show that Apollo outperforms cleaning solutions tailored to the respective applications. The design goal of Apollo is different. It is to provide a "quick and dirty" cleaning option that is suitable for a wide variety of applications. The main value proposition is to reduce the amount of data that needs to be processed by the application or by subsequent cleaning stages. We do compare Apollo performance to an *ideal* cleaning solution (i.e., to ground truth) as a proxy for highly-tailored application-specific cleaning solutions, in addition to the simple heuristics (e.g., voting and averaging). We show that Apollo approaches the former and outperforms the latter.

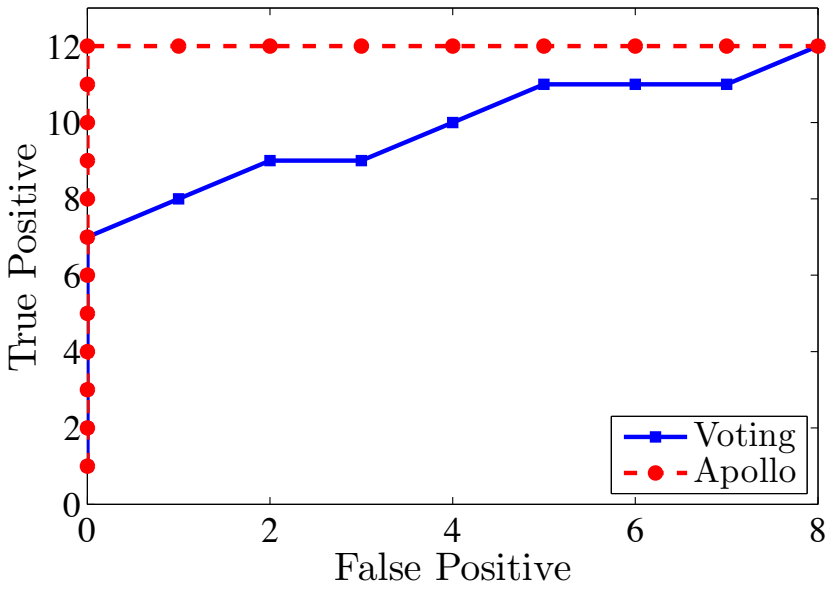


Figure 4.6: The ROC graph of correct observations in the simulated geo-tagging application.

Application I: Geo-Taggging

Geotagging applications are those where participants document locations of observations of interest. The observations themselves can be different across different applications. For example, locations of invasive species are collected in the "What's invasive" project [30]. A key challenge is to be able to tell true observations from erroneous or frivolous ones. Even when observations are supported by pictures, reported location tags can be inaccurate, which motivates some data cleaning.

In this section, we demonstrate a "toy" example of cleaning with Apollo, illustrating how it might be used in a simple application. To do so, we synthetically generate traces of 20 emulated hikers on a map, who observe occurrences of certain conditions on their path (for the sake of illustration, say, the hikers are asked to geotag locations of excessive litter). The ground-truth locations are chosen such that some lie on common paths whereas others are not well-visited. The emulated hikers report the coordinates of these locations if they come

	Apollo		Voting	
	Rel.	Irrel.	Rel.	Irrel.
Identified as Relevant	12	0	9	2
Identified as Irrelevant	0	8	3	6

Table 4.1: Comparing claim confusion matrices in the geotagging application for Apollo and voting scheme.

across them. The emulated reporting is not perfect. Some hikers are "less observant" (they miss reporting litter with some probability). Others report false observations.

In order to customize Apollo for this application, all one needs is a distance metric to indicate which claims are closer together. Since the claims merely report locations (of litter), the application-specific *distance metric* module is chosen to simply return the Euclidean distance between reported litter coordinates. Observations that are more than 50 meters apart are considered to be unrelated (and, hence, no link is generated between the corresponding claims by the distance module).

When Apollo completes its data distillation, eliminating the less credible locations of reported litter, it passes the rest on to the application that simply returns the surviving locations as the found locations of litter (actually, since location claims are clustered, the centroid of each cluster is returned).

We also compare Apollo to a voting-based scheme. In that scheme, location claims are clustered and clusters are ranked based on their size: clusters that are smaller than a threshold (i.e., have fewer votes) are dropped (i.e., deemed to be insufficiently corroborated observations). The rest are returned as true locations of litter.

We show, in Figure 4.6, that Apollo-based cleaning outperforms the voting-based scheme. We change the credibility threshold used in Apollo (for deciding which locations to return), as well as the threshold number of votes used in the voting scheme for the same purpose. We derive the number of true positives and false positives and plot them against each other, which is known as an ROC graph. An ROC graph is a common way of examining classifiers.

Ideally one wants to see a graph that is higher and closer to the Y-axis. It implies more true positives (i.e., fewer false negatives) as well as fewer false positives. A perfect classifier is one which includes the top left-most point in the plot, where the rate of false positives is zero and the rate of true positives is one.

The results show that Apollo, in this case, is an ideal classifier. It removes all false litter observations and reports all of the true ones. The voting based mechanism, on the other hand, either produces false positives or cannot report all relevant observations, for any chosen voting threshold.

Table 4.1 presents the *confusion matrices* for claims as classified (into true or false) by Apollo and the voting scheme. The confusion matrix shows the number of claims that have been classified correctly (as true or false) and the number that have been misclassified. For example, the right half of Table 4.1 shows that 9 actual event locations have been labeled correctly by voting while the remaining 3 actual location were misclassified as false. In contrast, Apollo classified all reports correctly.

Application II: PictureMe

PictureMe is a slightly modified geotagging application example, that allows individuals to take and share pictures of the objects they geotag. Consider an example campaign where participants are asked to pictorially document historic landmarks in their city. In this application, people are the sources. Pictures are the "claims" (claiming the photographed objects to be historic landmarks). Pictures of the same landmark share similarity, both in terms of location and visual content. The distance metric used in this application is therefore a combination of location and visual similarity. It allows similar pictures taken at nearby locations to be clustered together. When a large number of people participate in the collection, some sources might upload pictures of irrelevant objects (that are not true historic landmarks). While it may be possible to spot these by manual inspection, Apollo automates

such an inspection process. In this scenario, Apollo is used to distill the list of actual historic landmarks. Note that schemes that rely on corroboration alone (i.e., where a larger cluster of similar pictures gives more credibility to the content) will correctly report landmarks in more populated locations, but perhaps not those in places further off the populated routes. The latter may be captured by Apollo because it also computes source credibility (degree to which the sources appear to document true landmarks), and ranks pictures of credible sources higher, including those with a low degree of corroboration.

To evaluate the performance of Apollo in a controlled setting, we took pictures of historic campus landmarks that could, for example, be included in brochure of historic sites on campus. To design a controlled experiment, each source flips a weighted coin that depends on its pre-assigned (i.e., ground truth) probability of correctness. If the coin shows "true" the source takes a picture of an approved historic landmark from a given list. If the coin shows "false" the source takes a random picture. This is repeated multiple times for each source. Three data sets of pictures of selected landmarks and other sights were taken in this process (using camera-phones that upload the pictures and tweet the picture URLs). Each picture was geo-tagged with its associated GPS location.

As mentioned earlier, to customize Apollo to this application, we needed a corresponding *distance metric* module. We developed a module that uses GPS locations as well as a color correlogram (CC) [29] to establish logical distance between two pictures. The color correlogram measures the correlation between colors of pixels positioned a certain distance part. For a set of colors and distance values, we compute a feature vector for each picture. The logical distance between two pictures is given by a normalized sum of their physical distance and the Euclidean distance between two CC vectors. Physical distance is given a higher weight than color distance. Hence, pictures taken at different physical locations would have the highest logical distance (and be considered unrelated). Pictures taken at nearby locations that look different would have a lower logical distance. Finally, pictures

that look similar and are taken at nearby locations would have the smallest logical distance and be generally clustered together. After distillation, surviving pictures were passed to the application and presented to the user (e.g., for inclusion in the campus brochure).

The fraction of actual landmark pictures that survives distillation can be regarded as true-positives and the fraction of non-landmark pictures that are reported as well are treated as false-positives. In Figure 4.8, we plot an ROC graph to present the true positive rate against false positive rates for three different data sets. We also show results from voting that ranks clusters in descending order of size. We see that Apollo outperforms voting by a good margin using all different data sets. Figure 4.7 shows some of the top winning landmark pictures.

Figure 4.7: Some top winning landmark pictures, from left to right: Algelt Hall, Fallen Rocks at Main Quad, Tower at South Quad, Spring Colored Trees at Beckman Institute, Thinking Man statue at Main Library, Bike rail at Siebel Center

Application III: Speed Mapping

A major category of human-centric sensing applications involves people as sensor carriers, sharing time-series data from periodically sampled sensors. We implement an example traffic

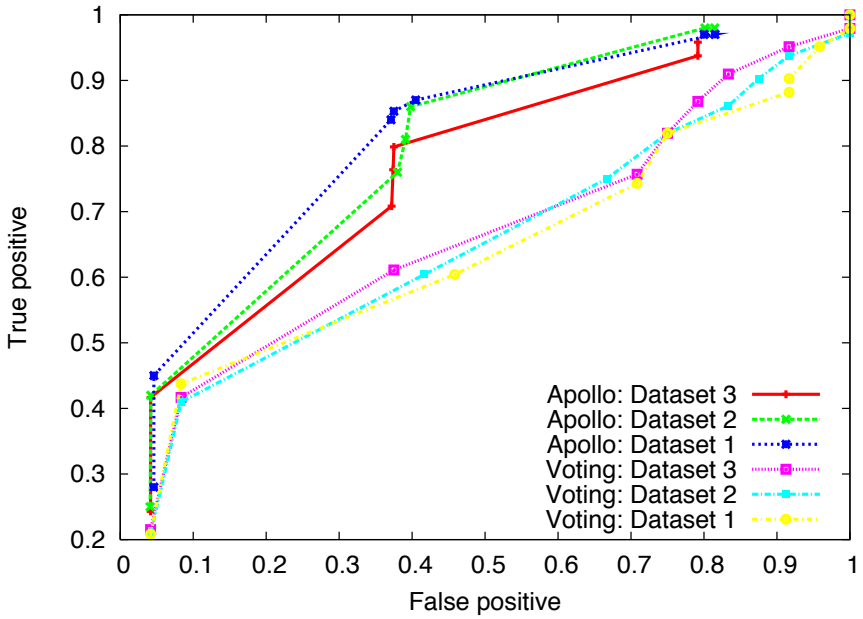
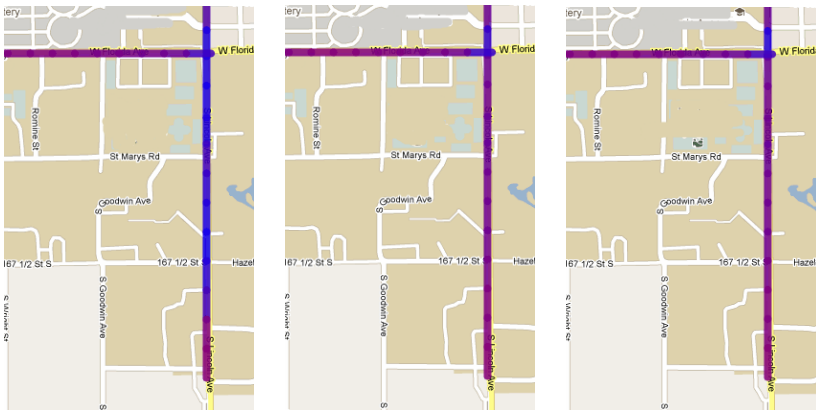


Figure 4.8: The ROC graph of the PictureMe application comparing Apollo with the voting scheme.

monitoring application to show how this type of data (namely, GPS trajectories in this case) can be cleaned using Apollo. While there are other techniques for cleaning time-series data, we include this case study in our evaluation to demonstrate the generality of Apollo and its ability to work well in cleaning vastly different types of inputs.

The goal of our speed mapping application is to create a speed map of different streets showing the average vehicular traffic speed as a function of location. Individuals carry phones that share GPS and speed information via Twitter. Apollo identifies and drops bad data. The rest are passed to the application and used to calculate the average traffic speed. We study the data cleaning performance in three different cases of injected data errors: (i) inconsistent context, (ii) faulty sensors, and (iii) incorrect calibration.

By inconsistent context we refer to cases where reported GPS trajectories do not represent those of a vehicle (instead, pedestrian data is mixed in). The erroneous inclusion of some pedestrian data in the mix alters the average speed values that are supposed to be computed for vehicular traffic. Several application-specific methods are proposed to detect



(a) Averaging (b) Apollo (c) GroundTruth

Figure 4.9: Output of the speed mapping application.

pedestrians and non-vehicles in speed mapping applications [31]. For example, accelerometer signatures can be used to distinguish pedestrians from drivers. We show how our application-independent framework can achieve similar data cleaning performance.

Faulty sensors are another common cause of error in all sensing applications. Noise removal often requires prior application specific knowledge such as a noise model or distribution of different variables, which (again) we do not need with Apollo.

Finally, incorrect calibration is a case where we misconfigure sensors to reports their data in different units (e.g., speed values in km/h instead of mph) unbeknown to the collection server. We use our Apollo-based implementation to remove such wrongly calibrated data.

Implementation We implemented an actual speed mapping application, where the client side runs on Android phones and reports its data to Apollo, whereas the server side takes its input from Apollo and computes the average traffic speed. Streets are segmented into 500-feet segments identified by a `segment_id`. Each claim consists of a `segment_id`, an average speed value and the number of samples used for averaging.

The *distance metric* is defined such that the distance is infinity if the `segment_id`s differ. Otherwise, it is the absolute difference in the speed value. After distillation with Apollo,

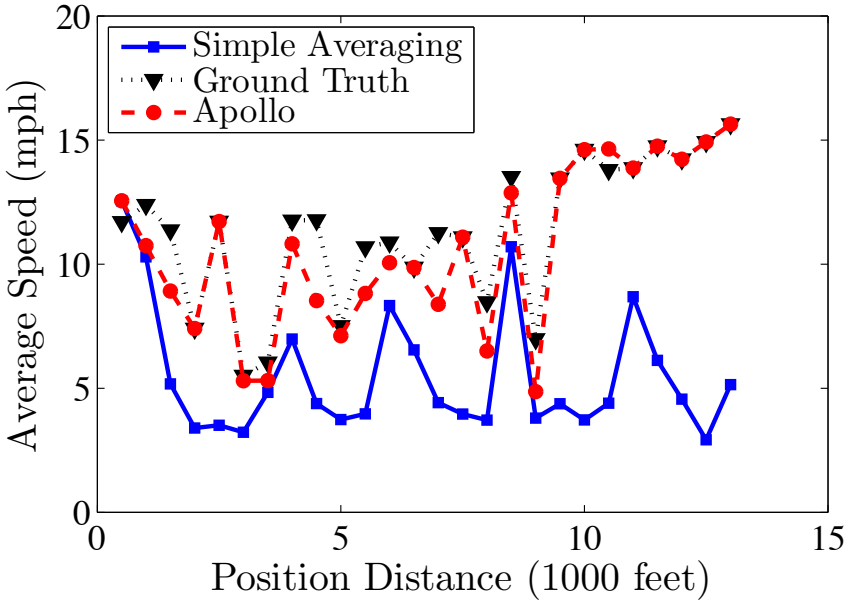


Figure 4.10: The average speed on Main street in the inconsistent context scenario.

the application takes surviving speed measurements and creates the speed map by simply averaging them for each road segment. A color-coded map of the area is produced using Google Map static API. Figure 4.9 is a map of an area that highlights the speed value at a particular location using color-coded marks. The deep blue color means zero speed and a complete red represents the maximum speed ($50mph$ in our experiments).

On the client side, GPS-equipped Android smart-phones (in particular Nexus One and Nexus S phones) do the data collection. An android application is developed to sample GPS location, time, speed, and bearing every 5 seconds. Two identifiers are added to each sample: a NodeID (the cellphone unique IMEI identifier), and a SampleID (cellphone local timestamp value). Each sample is then formatted as a string of key-values and shared via Twitter. A total of 15 hours of driving data was collected that covers 10 streets and around 180 miles.

Average Error (%)	Simple Averaging	Apollo
Main Street	41.89	9.23
Oak Street	7.71	6.17
First Street	6.89	6.38
Lake Street	0.0	0.53
All Streets	15.0	6.2

Table 4.2: Average error in the inconsistent context scenario.

Inconsistent Context In the first experiment with the speed mapping application, we investigate the ability of Apollo to remove data that are shared from an inconsistent context. Here, we insert sources who share GPS speed traces while walking instead of driving. When looking at the average speed value at each road segment for a particular street, we observe in Figure 4.10 that the lower pedestrian speed significantly reduces the average traffic speed values for the segments. However, our Apollo implementation of speed mapping removes the pedestrians from averaging and results in a curve much closer to ground truth. The ground truth represents the best possible performance of a well-designed application-specific scheme (e.g., [31]) .

Table 4.2 shows the average percentage error in speed estimation on four different streets. As the results suggest, using Apollo reduces the error compared to a simple averaging scheme.

Faulty Sensors In the second speed mapping experiment, we investigate the performance of Apollo in the presence of faulty sensors. This example naturally occurred during testing, as some of the phones (particularly Nexus Ones) produced very noisy speed values (in ranges from 0 to 150 miles per hour). Application-specific knowledge can be used here to remove the outliers from the data set. By doing this experiment, we aim to show how Apollo can achieve similar performance without using the application-specific knowledge. Again, we plot the speed curve for First street in Figure 4.11 based on both the ground truth and results from simple averaging and Apollo.

The application-specific noise removal scheme uses knowledge of speed limit values, re-

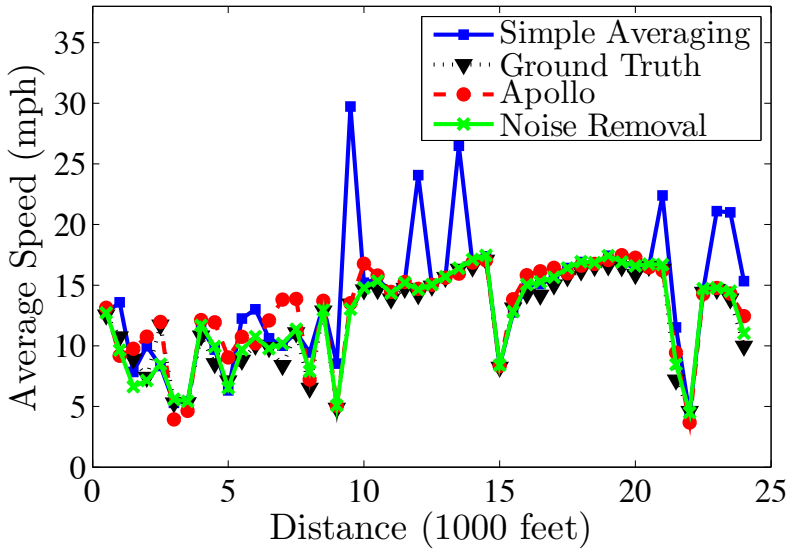


Figure 4.11: The average speed on Oak street in the faulty sensor scenario.

Average Error (%)	Simple Averaging	Noise Removal	Apollo
Main Street	21.76	8.95	14.04
Oak Street	13.36	5.56	5.29
First Street	25.25	5.32	16.18
Lake Street	29.55	1.54	2.4
All Streets	25.31	6.87	9.48

Table 4.3: Average error in the faulty sensor scenario.

moving samples that are larger than 30% above the street speed limit. We compare the average error over all street segments and report the results in Table 4.3. Note that, the result from Apollo is quite comparable with the application-specific noise removal scheme.

Incorrect Calibration The final experiment for the speed mapping application focuses on the case where the sensors are incorrectly calibrated. We emulate incorrectly calibrated devices by incorrectly reporting the speed in km/h instead of mph on some phones (without telling the right units to the receiver). Again, we compare both simple averaging and Apollo in computing the average speed values in each street segment. Figure 4.12 compares the color coded maps generated by each method. The higher speed values caused by converting

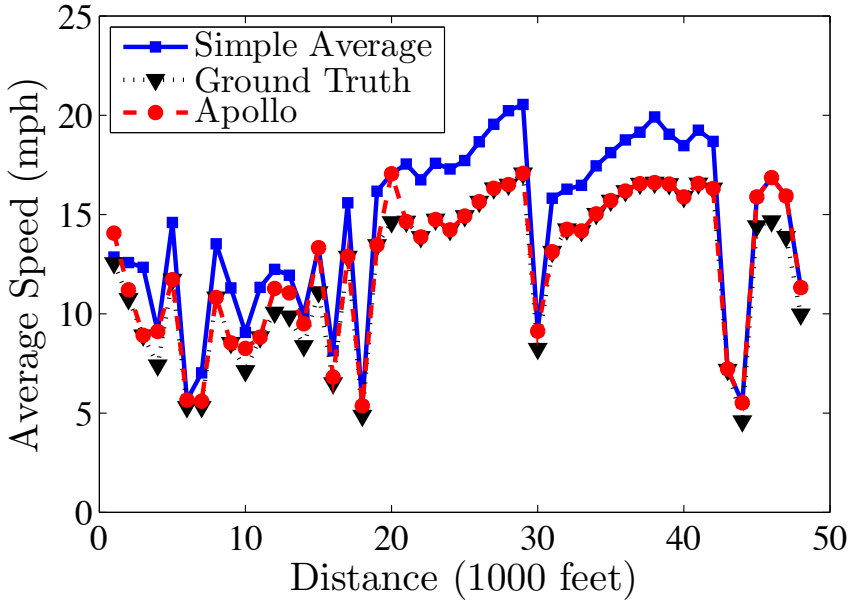


Figure 4.12: The average speed on First street in the calibration error scenario.

Average Error (%)	Simple Averaging	Apollo
Main Street	21.22	5.4
Oak Street	14.85	3.97
First Street	21.02	12.08
Lake Street	17.47	5.74
All Streets	19.02	6.63

Table 4.4: Average error in the incorrect calibration scenario.

mph to km/h are visible in the middle map. We show the average error over all segments in Table 4.4. The results show a significant improvement when using Apollo.

Application IV: Humans as Sensors

The third category of human-centric applications focuses on human as sensors. Social networks and in particular Twitter can be considered a huge source of human-generated sensing data. By allowing sharing only short messages (a.k.a "tweets") of no more than 140 characters, Twitter has become the main tool for sharing information in *real time*. As a result,

different search engines like Google and Bing already include tweets as an important part of their search results. In this application, we construct progression of historical events from collected tweets generated during the time of the real events.

Like other human-centric sensing applications there are several cleaning challenges involved. First, since anybody can share anything at anytime easily, a person can tweet about anything and anybody can produce bogus and irrelevant information. Second, it is not simple to figure out trust-worthiness of a source since anybody can create a Twitter account. Third, the amount of data produced on Twitter is massive, any algorithm has to be efficient enough to be practical.

Implementation Details The tweets were collected using Twitter API during the time of the real events (Cairo Unrest, Japan Tsunami, and London Riots). Despite the non-trivial nature of this application, building it was simple.

For the *distance metric*, we use the Jaccard distance [28], a simple yet widely used distance function for the content of the tweets. Jaccard distance is the ratio of the size of the set of shared keywords and the size of the set of all keywords of two tweets. All stop-words are discarded since they do not contribute any noticeable meaning to the tweets' content. Since a tweet is a relatively short string of text, we observed that Jaccard distance is sufficient for the purpose.

The application simply reports the most credible tweet of the hour every hour in chronological order. From this presentation, users are able to capture or reconstruct the history of events hour-by-hour.

Experiment setting We used more than 1 millions tweets from Egypt Unrest, more than 1 millions tweets from Japan Tsunami, and around 800 thousands tweets from London Riots (which happened during the course of this project). In all cases, the progression of events was successfully reconstructed hour by hour. The accuracy of the found facts were verified

with news reported by traditional media.

Egypt Unrest: In the case of Egypt Unrest, tweets were collected from February 1st until February 14th. We then chose 10 important events reported by media as ground truth. We compare the reported hourly top tweets to the chosen ground truth events. The results show that all ground truth events are in fact covered by the hourly top tweets returned by Twitter. We repeated the experiment with the voting scheme, returning only the tweet with the highest number of votes (i.e., number of sources) every hour. These tweets did not cover all the ground truth events. We then increased the reported tweets to top 2, 3, and so on. Using the voting algorithm, we eventually needed to look down to the 6th ranked tweet of every hour to find all of the ground truth events. This means that voting is not as good at ranking ground truth highly. While the difference between 1 and 6 might seems small as a ranking, the implication is that the user will need to wade through *six times more data* to find relevant information. Figure 4.13 shows the coverage of ground truth events (truth coverage) versus the lowest tweet tank one needs to consider to attain that coverage. Results are compared for Twitter and voting. Table 4.5 shows the ground-truth events and the found tweets by Twitter.

We then looked at ground truth facts reported by both Twitter and Voting, and compared the times that these facts were reported as the top tweet of the hour. Results show that Twitter reporting times are often earlier than those of voting. We suspect this is because voting has to wait until the fact reaches sufficient popularity. Table 4.6 shows the found facts and the reported time by Twitter, and voting (we could not track down media's exact reporting times).

Japan Tsunami: More than 1 million tweets were collected from March 11th 2011 to March 21th 2011. The above experiment was repeated for the new data set. As before, important events reported by media were used as ground-truth for evaluation. Table 4.7 compares 10 example events reported by media to the top tweets reported by Apollo showing great

Fact	Media	Tweet by Twitter
1	Google release speak2tweet technology for the people in Egypt	RT@googlearabia we are trying to spread these numbers among Egyptians: +16504194796 & +390662207294. Speak to Tweet. #jan25 #Tahrir Square
2	Number of protesters in Cairo's Tahir Square are revised to more than a million people	RT @AJELive: Al Jazeera's correspondent in #Egypt's Tahrir Square says that up to two million people are protesting in the square and surrounding areas.
3	Hosni Mubarak announce that he will on TV for a public address	RT @AJEnglish: Hosni Mubarak expected to speak to soon. Tune in to #AlJazeera to watch the coverage live: http://aje.me/ajelive #mubarak ...
4	Internet services partially restored in Cairo	FLASH: Egypt internet starts working in Cairo, other cities - users
5	Bursts of heavy gunfire early aimed at anti-government demonstrators in Tahrir leave at least five people dead and several wounded	RT @queen_iceis: Wow RT @bencnn: Witness in #Tahrir says pro-democracy people being shot at from rooftops, several dead. #Egypt #Jan25.
6	Hundred of thousands of anti-government protesters gather in Tahrir Square for what they have termed the "Day of Departure"	RT @sharifkouddous: Tahrir is getting packed. Ppl streaming in. They are calling today "The day of departure" for Mubarak #Egypt
7	The leadership of Egypt's ruling National Democratic Party resign, including Gamal Mubarack, the son of Hosni Mubarak. Hossam Badrawi, a member of the liberal wing of the party, became the new secretary-general	RT @BreakingNews: President Hosni Mubarak resigns as head of Egypt's ruling party, according to state TV - Sky News http://bit.ly/fHvJRr
8	Al Jazeera correspondent Ayman Mohyeldin is detained by the Egyptian military.	RT @DominiqueRdr: RT @evanchill: We can now tell you that our Cairo correspondent, @aymanM, has been in military custody for four hours. Please RT #Jan25
9	Ayman Mohyeldin is released seven hours later.	RT @bencnn: #AJE's @AymanM has been released! #freeayman
10	Wael Ghonim, a Google executive and political activist arrested by the state authorities since Jan 28 is released	RT @bencnn Wael @Ghonim has been released. #Tahrir #Egypt #Jan25

Table 4.5: Ground truth events and related tweets found by Twitter in Egypt Unrest

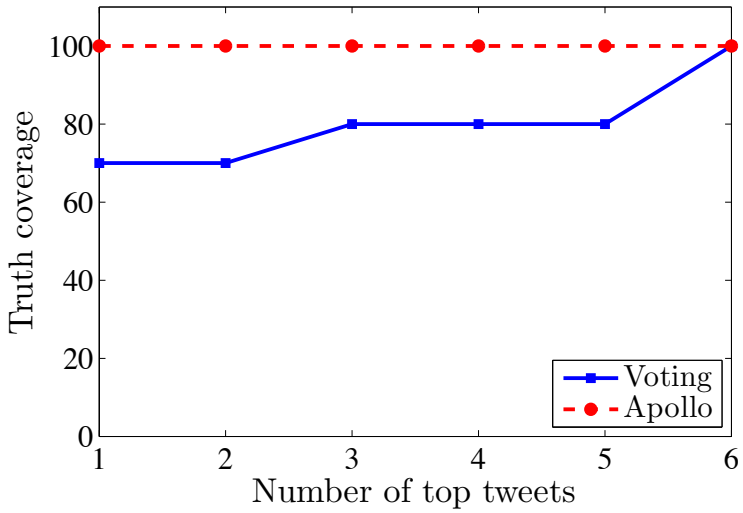


Figure 4.13: Truth coverage vs top results needed for Egypt Unrest dataset

correspondence between the two sets.

London Riots: Finally, around 800 thousands tweets were collected from August 9th 2011 to August 26th 2011 during London riots. Important events reported by media were used as ground-truth for evaluation. Table 4.8 highlights 10 such events and corresponding two tweets. It is clear in all cases that the top tweets reported by Apollo have great correspondence to events reported later by media (which we consider ground truth).

The above is of course not a conclusive evaluation of Apollo performance, but rather anecdotal evidence obtained from several different data sets that suggests the efficacy of data distillation at retaining the most important events after the distillation process.

The results show that the hourly top-1 tweets of Twitter alone cover the important ground truth facts; while in the voting algorithm, some facts were ranked lower. Figure 4.14 shows the true event coverage versus the number of tweets that need to be retained from the top in order to achieve the coverage in question for the Japan data set. The ranking of Apollo and voting schemes are compared. It can be seen that Apollo ranks these events higher, meaning that a user will need to wade through less data to find relevant facts. A similar favourable

Fact#	Twitter	Voting
1	2011-02-01 11:00	2011-02-01 11:00
2	2011-02-01 13:00	2011-02-01 15:00
3	2011-02-01 19:00	2011-02-01 20:00
4	2011-02-02 10:00	2011-02-02 11:00
5	2011-02-03 02:00	2011-02-03 04:00
6	2011-02-04 07:00	2011-02-04 08:00
7	2011-02-05 16:00	2011-02-05 16:00
8	2011-02-06 14:00	2011-02-06 16:00
9	2011-02-06 20:00	2011-02-06 20:00
10	2011-02-07 15:00	2011-02-07 16:00

Table 4.6: Report time of ground-truth facts by Twitter and voting. The rounded times are due to the fact that we report top-1 tweets of each hour

result was also observed in the London Riots dataset.

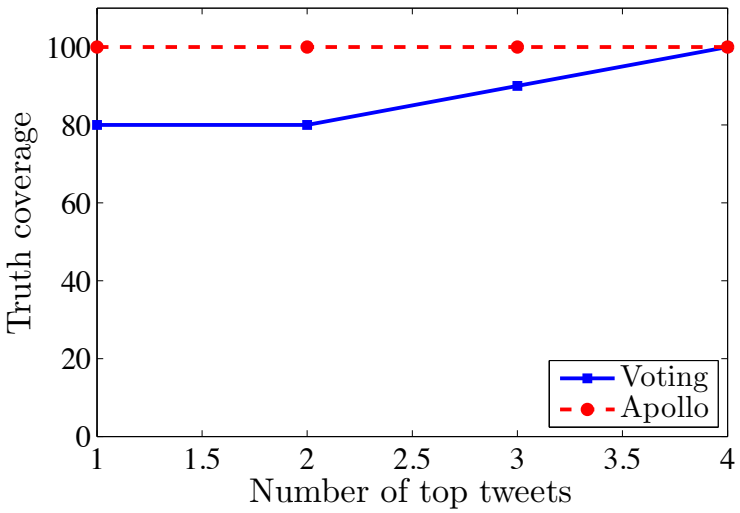


Figure 4.14: Truth coverage vs top results needed in Japan Earthquake dataset

Latency: The latency observed between Twitter and voting are too closed to report in this case.

Effect of Quality of Distance Function In the Timeline Recovery application, we the text distance function is a jaccard distance as explained. In our implementation, we filtered out words which do not contribute to the meaning of the tweets (which called stop-words).

Fact	Media	Tweet by Twitter
1	The first earthquake hit Japan	RT @HuffPostWorld: BREAKING: Massive 7.9 earthquake reportedly rocks Japan, 19-foot tsunamis feared http://huff.to/ezzmQb
2	The government warns the possibility of radiation leak	RT @Reuters: Japan warns of radiation leak from quake-hit plants http://t.co/iAFcDZg
3	Large number of dead and missing were reported	RT @BreakingNews: Latest Japan quake toll: 398 dead, 805 missing - Kyodo
4	Prediction of high probability of nuclear meltdown at Fukushima	RT @Reuters: FLASH: #Japan nuclear authorities say high possibility of meltdown at Fukushima Daiichi No. 1 reactor - Jiji
5	Explosion at Fukushiman nuclear plant	RT @BreakingNews Explosion heard at quake-hit Fukushima nuclear plant in Japan - AFP via Sky News ¡= oh gosh :/
6	Another big earthquake hit Fukushima	Not again RT @BreakingNews: Quake with preliminary magnitude of 6 hits Fukushima in northern Japan - Reuters via NHK
7	Cooling system at Fukushima nuclear plant failed	RT @komonews: RT @Reuters: Japan's nuclear safety agency says Fukushima Daiichi Nuclear Plant No. 3 reactor's emergency cooling system not functioning
8	Many other quakes reported hit Japan.	RT @cnnbrk: USGS: More than 140 quakes mag. 4.5 and higher in NE Japan in 24 hours #quake http://on.cnn.com/fyrSuV
9	Another reactor at Fukushima has problem	RT @BreakingNews: Japan's nuclear safety agency reports an emergency at a second reactor - AP http://bit.ly/hk8r1u
10	Nuclear meltdown was suspected to happen at Fukushima	RT @GeorgeTakei: RT @CNN: Meltdown may be under way at Fukushima nuclear reactor, an official w/ Japan's safety agency says. #NowWePray

Table 4.7: Ground truth events and related tweets found by Twitter in Japan Tsumani

Fact	Media	Tweet by Twitter
1	Information about London riot flood social media sites	London Riots: Twitter Traffic Surges in the UK [STATS]: Traffic is surging to Twitter and major media website... http://bit.ly/qh45N6
2	Blackberry's website was hacked, which made speading of riot worse	Hackers hit Blackberry over riots: A hacker group has attacked Blackberry's website after the company said it wo... http://bbc.in/nz2hJ7
3	UK government want Facebook, Twitter, and RIM to notice about their role in the riots	UK Govt. to meet Facebook, Twitter and RIM about their responsibility to not fuel riots http://t.co/Vm1xJCp via @thenextweb
4	A young woman was arrested to using Blackberry messenger to aid the riots	18-year-old woman from east London charged under Serious Crime Act for using Blackberry messenger to encourage others to take part in #riots
5	Riots got worse in Hackney	#tools Rellen in Londen HQ London Riots Now In Hackney: KIJK VERDER: urly.nl Zorg dat je bo... http://t.co/f6PeWCt #internetmarketing
6	Police started to use tougher solutions	UK riots: Police water cannon and plastic bullets? After 50 years of the most lavish welfare state on earth? Wha... http://t.co/6Qv24hk
7	The priminister critize the "broken society"	Cameron blames UK riots on 'moral collapse' http://bit.ly/pmOuLc
8	The priminister consider the posibility of blocking social media sites to avoid spreading of the riots.	UK's Cameron Ponders Blocking #social #media Sites if Riots Continue http://t.co/qiEQdNIV
9	Facial Recognition Software is started being used to identify suspects appeared on social media sites and surveillance systems.	Police Used Facial Recognition Software To ID Suspects in UK Riots [Riots] http://t.co/rKbxxEG
10	Police cracked important plot attacking sensitive venues.	UK police say they foiled planned attacks on 2012 Olympics site, shopping centers, in riots -AP #LondonRiots

Table 4.8: Ground truth events and related tweets found by Twitter in London riots

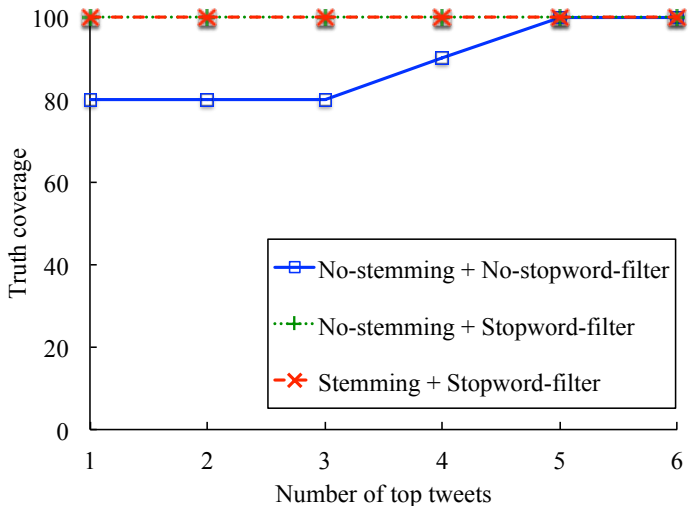


Figure 4.15: Effect of Quality of Distance Function

In this part, we try different versions of the distance function with different add-ons. An add-on that we consider beside filtering stop-words is stemming. The idea of stemming is to reduce the words to their root form. For examle, "protesting" shall be reduced to "protest"; there for two tweets with different forms of a same word will beconsided as sharing that word. Figure 4.15 show the performance of Apollo in the Timeline Recovery application with different version of the text distance functions. As we observe, stop-word filtering does improve the performance of Apollo, while stemming is not neccessary. This hint us that as long as the distance function is sufficiently refect the coarse-rgdifference

4.3.2 Discussion

Our work confirmed the hypothesis that it is possible to build (largely) application-independent data cleaning services for social sensing applications that can distill different data formats including boolean observations, time-series data, as well as unstructured data such as text and images. There is a possibility of using the links between different content types (like image and text report the same fact about an event) to enrich the information in the network of sources and claims, but this work by its own is a sizeable problem. We

are having some research collaboration along that direction, but it is too early to report any confident result; we, therefore, leave this as a future work.

Many interesting aspects of designing a good data distillation service remain candidates for future work. Importantly, we have not investigated the robustness of the service to malicious use. In principle, if the algorithm used to determine the credibility of claims is made known, it becomes possible for colluding agents to foil it; a topic we have not addressed. Individuals can also gain credibility by consistently providing true observations, then exploit that accumulated credibility to insert bad data into the mix and have it pass the distillation filter. This problem is common to reputation-based systems. It remains relevant here since the underlying algorithm tries to estimate source credibility as well.

Our focus has been on describing a general architecture for data distillation. Once the architecture is defined, it is possible to improve the underlying maximum likelihood estimation algorithm as needed. The algorithm described in this chapter does have several limitations. For example, we have not explicitly described how to handle conflicting claims. We also have not described how to account for non-independent sources. In the case of Twitter, it could be that a large set of individuals report the same observation not because they independently observed themselves it but because they heard it from a source they trust (which could in fact be wrong). Such rumor propagation needs to be detected, possibly by observing correlations in reporting among sources over time, or by considering the social network of sources. Once an appropriate solution to this problem is adopted, it can be easily incorporated into our architecture simply by upgrading the maximum likelihood estimation algorithm to account for the additional information.

We illustrated the use of Apollo in the context of many different applications. In many cases, other application-specific outlier elimination techniques could have been used. We stress the word *application-specific* because, for example, the numeric techniques that could have been used for outlier detection in ordered time-series data might not necessarily work

for text and pictures. The contribution of Apollo therefore lies in laying a foundation for data distillation that works well across a large spectrum of dissimilar input types.

Till this point, we presented Apollo, a general data distillation service for human-centric sensing. The service cleans input data in scenarios where data are noisy, sources are unreliable, or their reliability is unknown. As a general service, Apollo was designed to operate with minimum application dependent parts and provide a first tier of data cleaning to reduce the data processing burden at later stages of the workflow. As demonstrated, Apollo effectively reduced bad data in several representative scenarios of human-centric participatory sensing that span text, images and time-series data.

CHAPTER 5

SOURCE SELECTION IN SOCIAL SENSING APPLICATIONS

The study in this chapter develops algorithms for improved source selection in social sensing applications that exploit social networks (such as Twitter, Flickr, or other mass dissemination networks) for reporting. The collection point in these applications would simply be authorized to view relevant information from participating clients (either by explicit client-side action or by default such as on Twitter). Social networks, therefore, create unprecedented opportunities for the development of sensing applications, where humans act as sensors or sensor operators, simply by posting their observations or measurements on the shared medium. Resulting social sensing applications, for example, can report traffic speed based on GPS data shared by drivers, or determine damage in the aftermath of a natural disaster based on eye-witness reports. A key problem, when dealing with human sources on social media, is the difficulty in ensuring independence of measurements, making it harder to distinguish fact from rumor. This is because observations posted by one source are available to its neighbors in the social network, who may, in-turn, propagate those observations without verifying their correctness, thus creating correlations and bias. A corner-stone of successful social sensing is therefore to ensure an *unbiased sampling* of sources that minimizes dependence between them. This chapter explores the merits of such diversification. It shows that a diversified sampling is advantageous not only in terms of reducing the number of samples but also in improving our ability to correctly estimate the accuracy of data in social sensing.

5.1 Introduction

This chapter investigates algorithms for source selection in social sensing applications[3]. We interpret social sensing broadly to mean the set of applications, where humans act as the sensors or sensor operators. An example application might be a participatory sensing campaign to report locations of offensive graffiti on campus walls, or to identify parking lots that become free of charge after 5pm. Another example might be a damage assessment effort in the aftermath of a natural or man-made disaster, where a group of volunteers (or survivors) survey the damaged area and report problems they see that are in need of attention. Social sensing benefits from the fact that *humans* are the most versatile sensor. This genre of sensing is popularized by the ubiquity of network connectivity offered by cell-phones, and the growing means of information dissemination, thanks to Twitter, Flickr, Facebook, and other social networks.

Compared to applications that exploit well-placed physical sensors, social sensing is prone to a new type of inaccuracy; namely, unknown dependence between sources, which affects data credibility assessment. This dependence arises from the fact that information shared by some sources (say via a social network such as Twitter) can be broadly seen by others, who may in turn report the same information later. Hence, it becomes harder to tell whether information received is independently observed and validated by the source or not. When data items are inherently noisy, one would like to use the degree of corroboration (i.e., how many sources report the same data) as an indication of trustworthiness. For example, one would like to believe an event reported by 100 individuals more than an event reported by a single source. However, if those individuals are simply relaying what they heard from others, then the actual degree of corroboration cannot be readily computed, and sensing becomes prone to rumors and mis-information.

[3]This work was done in collaboration with Md Yusuf S Uddin, Md Tanvir Al Amin, Tarek Abdelzaher, Boleslaw Szymanski, and Tommy Nguyen

This chapter investigates the effect of diversifying the sources of information on the resulting credibility assessment. We use Twitter as our social network, and collect tweets representing events reported during Egypt unrest (demonstrations in February 2011 that led the resignation of the Egyptian president) and hurricane Irene (one of the few hurricanes that made landfall near New York City in 2011). In our dataset, some of the tweets relay events that are independently observed by their sources. Others are simply relayed tweets.

While it is generally impossible to tell whether or not content of two similar tweets was independently observed, our premise is that by analyzing the social network of sources, we can identify those that are "close" and those that are "not close". By using more diversified sources, we can increase the odds that the chosen sources offer independent observations, and thus lower our susceptibility to rumors and bad information.

The chapter explores several simple *distance metrics* between sources, derived from their social network. Distance may depend on factors such as whether one source is directly connected to another (e.g., one *follows* the other in Twitter lingo), whether both are connected to a common ancestor (e.g., both follow a common source), or whether both are followed by the same people. By choosing the most dis-similar sources, according to these metrics, we show that we can indeed suppress more rumors and chain-tweets. The impact of different distance metrics on improving credibility assessment is compared.

The rest of this chapter is organized as follows. Section 7.2.2 describes a set of earlier works done in field of source selection and fact-finding. Section 5.2 formulates the source selection problem and then proposes a set of source selection schemes that diversify the collection of sources passed to the credibility estimator. Evaluation results demonstrating the effect of source selection in credibility assessment are presented in Section 5.4 followed by conclusion and future research direction.

5.2 Source Selection in Social Sensing

Social sensing applications that exploit social networks (e.g., Twitter) allow users to report events that are not entirely experienced or verified by themselves. This is because individuals are able to reproduce claims that they heard from others. We argue that if information can be collected from a diverse set of sources who have a weak "social" connection between them, there is a higher chance that the information collected thereby would be more independent, allowing a more informed judgment to be made regarding its reliability. This motivates diversifying source selection in social sensing. In the following, we use the terms users, sources and nodes as well as the terms tweets, feeds, claims and observations interchangeably.

5.2.1 Online User Social Graph and Source Dependence

In any online community platform or online social network, each user maintains a virtual relationship with a set of other users. This relationship entails some degree of information sharing. For example, on YouTube, a user may subscribe for videos posted by another user so that the former gets a notification when the later uploads a new video. In Facebook, there is an explicit friend relationship and a membership of a fan-page of another well-known user. Google$^+$ has more granularity like friends, family members, acquaintances, and other groups, called circles. In this chapter, we consider a Twitter-based social sensing application, which allows a *follower-followee* relation. A user following another user means that the former intends to receive the posts made by the latter. We say that if user i follows user j, i is the follower and j is the followee. In Twitter, a user can arbitrarily choose which other users to follow, although the converse is not true. That is, a person can not make another user to follow them (a person can, however, block another user from following).

We leverage this relationship in Twitter to form a *social graph* among users. We represent each user by a vertex in the graph. A directed edge from one vertex to another denotes that

the latter follows the former. We use the notation $i \rightarrow j$ to denote an edge in the graph meaning that user i follows user j. Sometimes, a user may not directly follow another, but can follow transitively via a set of intermediate followees. We refer to this as a follow chain. We use $i \rightarrow^k j$ to denote such a chain with k edges in between. Obviously, $i \rightarrow j = i \rightarrow^1 j$. If i follows j via more than one path, $i \rightarrow^k j$ designates the one with the least number of hops. We also use $F(i)$ to denote the set of users that a node i follows, that is, the set of followees of node i.

It is reasonable to argue that if source i directly follows source j, reports posted by j would be visible to i, making the information posted by i potentially not original. Another possibility could be that both source i and j have another source in common that both of them follow (i.e., they have a common followee). In that case, the common followee may impact both of them, making their observations mutually dependent. In order to extract reliable information from user-generated tweets, our intention is to gather tweets from *independent* sources to maximize the odds of originality of the information (or equivalently minimize the chance that these users influenced one another in making their tweets). The question is how to reduce potential dependence among users as a function of the aforementioned relationship between them. In the following, we formulate the source selection problem.

5.2.2 Source Selection Problem Formulation

We construct a *dependence graph* consisting of sources as vertices and directed edges between vertices as an indication whether or not a source is potentially dependent on another source (e.g., receives their tweets). Weights assigned to edges reflect the degree to which such influence can happen. These weights depend on the characteristics of the social network and the underlying relationship among sources in the social graph. In the context of Twitter, we simply use the follow relationship between sources. If we consider the follow relationship to be the only way sources could be dependent, the proposed dependence graph is identical

to the Twitter social graph itself. In general, it is reasonable to assume that other forms of dependence may also exist.

Let $\mathcal{G} = (V, E)$ be the dependence graph, where an edge ij indicates source i is potentially dependent on j. Each edge ij is assigned a *dependence score*, f_{ij}, that estimates the probability of such dependence. That is, with probability f_{ij}, source i could make the same or similar claims as source j. Many factors affect these dependence scores. For example, when a source directly follows another source it is more dependent on its followee than a source that follows the same followee via a longer follow chain. The number of common followees between a pair sources can also be an indication of dependence between them. If a given pair of nodes have a large number of common followees, they are prone to be more dependent than a pair that have fewer common followees or no followees at all. Whatever the cause of dependence between sources is—that we describe in the subsequent subsection in more detail—we aim to choose a subset of sources that have the least amount of dependence among them.

Without loss of generality, we can assume that the dependence graph, $\mathcal{G}$, is a complete graph. That means, f_{ij} exists for every pair of sources i and j (f_{ij} can be assumed to be zero if no influence exists between the corresponding sources). We are interested in estimating to what extend a source can make an *independent* claim, although its claims can be influenced by those made by others. We define an overall *independence score* for each source that gives the probability that it is *not* influenced by other sources in making a claim. This score, denoted by $\beta(i)$ for source i, can be defined as:

$$\begin{aligned}\beta(i) &= P[i \text{ is independent in making claims}] \\ &= \prod_{j=1}^{n} P[i \text{ is not dependent on } j] \\ &= \prod_{j=1}^{n} (1 - f_{ij}) \end{aligned} \tag{5.1}$$

One important property of independence score (we shall henceforth refer to as β-score) is that a source cannot have this score in isolation. It is rather a functional form of dependence on other sources. From the definition, we observe that $\beta(i) = 1$ means that source i is absolutely independent (not dependent on any other sources in consideration). We also notice that the β-score declines for a source if the source is influenced by more other sources. To diversify the collection of sources, we consider only a subset of sources whose sum of independence scores is maximum subject to the constraint that no individual source has an independence score below certain threshold. Let this threshold be τ. That is, we want to compute the subset of selected sources $S \subseteq V$ that maximizes the sum of β-scores. Therefore, we have:

$$\max \quad \sum_{i \in S} \prod_{j \in S} (1 - f_{ij}) \tag{5.2}$$

$$\text{s.t.} \quad \prod_{j \in S} (1 - f_{ij}) \geq \tau, \forall i \in S \tag{5.3}$$

Note that, individual sources can also have some kind of *influence* factor associated with them that can be inferred from the number of followers. If a source has many followers, it may mean that this source produces observations that other users find reliable. This is a source ranking problem and has been addressed in prior work. In this dissertation, we do

not address source ranking. Instead, we verify the promise that *diversifying* the sources can improve the performance of a subsequent ranking algorithm.

The optimization problem stated by Equation 5.2 can be shown to be an IP (Integer Programming) problem, and is therefore NP-Hard. We can use a greedy approximation by building the solution incrementally. The greedy algorithm assumes that all candidate sources are available apriori so that the source selection can pick a subset of them. Sometimes the set of sources is not known beforehand. Rather, new sources are discovered as they arrive incrementally. In that case, an *online* algorithm seems more appropriate.

We consider a system where a stream of tweets arrives at a processing station. Our source selection scheme acts as an admission controller that needs to make an online assessment regarding whether or not a new source is to be selected based on the relationships it has with respect to other sources selected earlier. If the source is selected, all tweets originated from that source are admitted, and will be passed to the actual processing engine as they arrive. Otherwise, the source is not admitted and all tweets from that source will be dropped. Hence, our online admission controller is a simple gate that admits tweets based on which source they are coming from. An advantage of admission control as described above is that it is fast and easy. In particular, it is based on sources and not on the content of tweets. In principle, better admission controllers can consider content as well, but they will be significantly slower. Hence, we restrict our notion of data sampling to the granularity of entire sources, making it a source selection scheme. In the following, we compare performance of different source selection schemes.

5.3 Online Admission Control

The online admission controller makes a decision regarding each tweet upon its arrival to the system. If the source associated with the tweet is already admitted, the tweet is passed to the

next step. If not, the candidacy of the source is evaluated in terms of how independent this source is with respect to the earlier admitted sources. The admission controller computes the β-score of the incoming source and then accepts it only if its β-score remains above an admission threshold, τ. Otherwise, it is denied. Let S be the set of sources that have been admitted so far. The source denial rule, as per Equation 5.3, is:

$$\textit{Denial rule for source } i\text{:} \quad \prod_{j \in S} (1 - f_{ij}) < \tau \tag{5.4}$$

For a certain definition of f_{ij} and the associated admission threshold, τ, we can formulate a set of different admission controllers as we describe in the following. In all admission control schemes, if not otherwise stated, admission decisions are final: once admitted, a source is not revoked from the admitted set. In the following discussion, let i be the source who is seeking admission.

1. No direct follower:

$$f_{ij} = \begin{cases} 1 & \text{if } i \text{ follows } j \\ 0 & \text{otherwise} \end{cases}$$

$$\tau = 1$$

Deny, if the source is a direct follower of another admitted source. Recall that if source i follows any of the earlier admitted sources in S, that is, for some $j \in S$, $f_{ij} = 1$, it leads to $\beta(i) = 0$, thus violating the admission condition.

2. No direct follower as well as no common followee:

$$f_{ij} = \begin{cases} 1 & \text{if } i \to j \vee F(i) \cap F(j) \neq \emptyset \\ 0 & \text{otherwise} \end{cases}$$

$$\tau = 1$$

Deny, if the source directly follows someone in the set or has at least one followee in common with another admitted source.

3. No descendants:

$$f_{ij} = \begin{cases} 1 & \text{if } i \to^k j, 0 < p < 1 \\ 0 & \text{otherwise} \end{cases}$$

$$\tau = 1$$

Deny, if the source is a follower of another admitted source possibly via a set of intermediate followees.

4. β-controller: This controller selects sources that progressively improve the sum of β-scores as per Equation 5.2, while satisfying the constraint 5.3 for each individual admitted source. This controller treats transitive *follower-followee* relationship among sources and defines the following dependence function:

$$f_{ij} = \begin{cases} p^k & \text{if } i \to^k j \\ 0 & \text{otherwise} \end{cases} \tag{5.5}$$

for some constant $p < 1$. We used, $p = \frac{1}{2}$.

Let $B(S)$ be the sum of β-scores of admitted sources, i.e, $B(S) = \sum_{j \in S} \beta(j)$. Let i be the new source. The scheme computes:

$$\beta'(i) = \prod_{j \in S \cup \{i\}} (1 - f_{ij}), \forall i \in S \cup \{i\} \tag{5.6}$$

$$B(S) = \sum_{j \in S} \beta(j) \tag{5.7}$$

$$B'(S) = \sum_{j \in S \cup \{i\}} \beta'(j) \tag{5.8}$$

The scheme then admits i only if $\beta'(i) \geq \tau$ and $B'(S) > B(S)$. Note that, when a new source is admitted, the scores of some earlier admitted sources may decrease (this is because they may be followers of this newly admitted source). Upon admittance of the new source, those scores are updated. Among possible choices, we consider two versions of β-controllers, with $\tau = 0, 1$. The one with $\tau = 0$ does not check individual β-scores but admits sources as long as they improve $B(S)$, whereas $\tau = 1$ denies a new source if it has any link with any of the earlier admitted sources (i.e., $\beta < 1$) and also fails to improve $B(S)$.

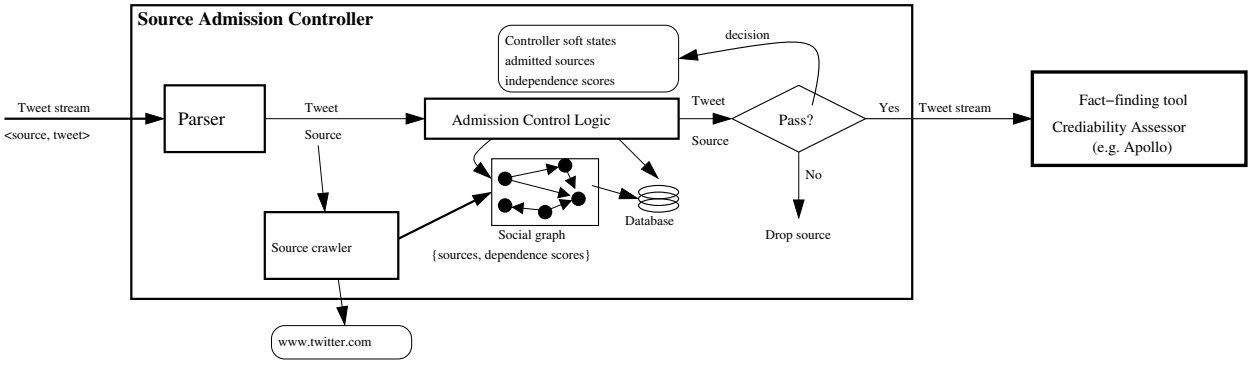


Figure 5.1: Schematic model of the admission controller with Apollo's pipeline.

Complexity of Admission Controllers: Once accepted, a source is not rejected later, and

vice versa. So the decision about a particular source can be stored in a hash table. Once a source arrives, whether that source had already been explored or not, can be checked in $O(1)$ time and the stored decision can be used. If the incoming node is previously unexplored, the admission controller needs to decide about it. For controllers 1 to 3, this decision requires $O(out(i))$ computations, where $out(i)$ is the outdegree of i in the dependence graph. The method is simply to check whether any of those outdegree vertices belong to the set of already decided sources. β-controllers consider ingoing edges also, so it takes $O(out(i)+in(i))$ computation per admission decision. In short, admission cost of a new source is at worst in the order of its degree in the dependency graph. But it is O(1) lookup for all the tweets that come from it thereafter. Moreover, social graphs tend to have a power law degree distribution, so very few nodes will require a high computation time for the decision.

5.3.1 System Design and Implementation

Our admission controller is used in association with a fact-finding tool or a credibility assessment engine. The engine tries to extract summary information of high quality from a pool of many unreliable claims. Apollo [32] is the engine used. Apollo receives a stream of tweets from which it derives credibility scores of sources and claims (i.e., tweets). Given a source-claim matrix containing which source makes which claims, Apollo uses an expectation-maximization (EM) technique [33] that iteratively assigns truth values to those claims and correctness probabilities to sources, so that the expected likelihood of these assignments is maximized. Once the iterations converge, Apollo outputs the top credible sources and top credible tweets made by those sources.

Apollo assumes that all sources are independent. Our admission controller filters out tweets before they are fed into the Apollo engine such that the surviving ones are more likely to be independent indeed. Figure 5.1 shows the design of the whole pipeline.

The pipeline is implemented as a set of stages processing a stream of tweets in JSON

format. A parser extracts various information components from each tweet entry. There are two main components to extract: user information, usually a unique Id and screen name of the source who tweeted the current tweet, and the tweet string itself. The admission controller maintains a source information base that is updated as it encounters new sources. Upon encountering a new user, the "source crawler" contacts to the twitter server and collects the Twitter record of that particular user, which includes additional information such as the user's screen name, location, profile url, the number of followers and the number and identities of followees this user has. If not otherwise restricted by any privacy setting for this user, the crawler also collects the complete list of followees (i.e., the other users that this user follows in twitter's user space). As more and more sources are encountered, a social graph among users is constructed. This social graph is stored in a database and is an essential element for source admission control.

An admission controller logic unit implements the admission control rules described in Section 5.3. It computes dependence scores between pairs of sources and admits new sources as permitted by the corresponding admission rules. When an incoming source is admitted, the associated tweet entry is passed to the next processing stage within Apollo.

5.4 Evaluation

We evaluated our source selection schemes using two twitter datasets. One is for Egypt unrest, collected in February 2011, during a massive public uprising against the ruling government in Cairo. Another dataset is from hurricane Irene, one of the costliest hurricanes on record in the Northeastern United States, collected in August 2011, when it made landfall near New York City. In both cases, we collected thousands of tweets as posted or shared by online users as the events unfolded during those times (Table 5.1. We were interested in extracting a smaller subset of high quality information on the progress of these events as computed by the find-finder engine, Apollo. The question is whether a significant improvement occurs in distilling the most credible tweets due to the source diversification process described earlier in this chapter.

Table 5.1: Statistics of two datasets

Dataset	Egypt unrest	Hurricane Irene
Time duration	18 days	$\approx$ 7 days
# of tweets	1,873,613	387,827
# of users crawled	5,285,160	2,510,316
# of users actually twitted	305,240	261,482
# of follower-followee links	10,490,098	3,902,713

In Twitter, both the number of followers and followees per user observe a power law distribution (i.e., heavy tail distribution). More precisely, there exists a very large number of users who have only a few followers, whereas a few sources may have an extremely large number of followers. The same is true for the number of followees. Figure 5.2(a) plots the complementary cumulative distribution (CCDF) of the number of followers and followees per source across all users recorded in the Egypt dataset and Irene dataset. The CCDF depicts what fraction of users have the number of followers or followees greater than a given number appearing in x-axis.

In Figure 5.2(a), we observe that the number of followers per user, in both datasets, is

larger than the number of followees per user. This is why the followee curve in the plot lies beneath the follower curve. That means for a given number, x, the number of users having follower count equal to or greater than x is larger than the number of users having the same number of followees or more. It is quite natural in Twitter. Usually for most users, the number of followers is greater than the number of followees. To illustrate this, we also plot the ratio of follower count to followee count (*ff-ratio*) in Figure 5.2(b). We see that in both datasets only a very small fraction of users have non-zero follower and followee count (1.7% for Egypt dataset and 2.4% for Irene dataset). And nearly 1% users have more followers than followees (ff-ratio > 1). Moreover, a very few of them have a magnitude order of more followers than followees. These are mostly popular entities, such as celebrities, international organizations, and news media sources.

Next, we present results from various admission controllers that we described in Section 5.3. We compare no admission control, *no follower* (No FLWR), *no common followee* (No CF) and *no descendant* (No DT), and β*-controller* (Beta). Our two datasets have two distinct properties. For instance, the Egypt dataset contains a large number of users who are connected by links (they have follower-followee relationship). To demonstrate the promise of independent sources, we *clean* this dataset by eliminating tweets from all those users who do not have any link with any other sources. That means socially isolated sources are not considered. This is important because these sources are trivially independent, hence are admitted by almost all admission schemes. However, we were not able to do that same for Irene dataset, because doing so leaves us only a few users (2,952 out of 261,482). We instead keep the Irene dataset as it is and demonstrate the performance of our admission controllers for the case when there is less connectivity in the underlying social network. Conceptually, our admission controllers, by their very design, exploit links between sources to choose independent sources, if there is any; otherwise, they remain indifferent to sources and do no harm to the usual operation.

We evaluate the improvement in performance attained by these admission controllers in Apollo's ability to select credible tweets. We assess exactly what fraction of tweets were "good" in that they reported true facts (as determined by human ranking). This fraction defines the quality of the result. Hence, once Apollo returns the top tweets, we ask volunteers to grade these tweets by placing them in one of the following two categories:

- *Fact*: A claim that describes a physical event or an instance that is generally observable by many individuals *independently* and can be corroborated by sources external to the experiment (e.g., news media).

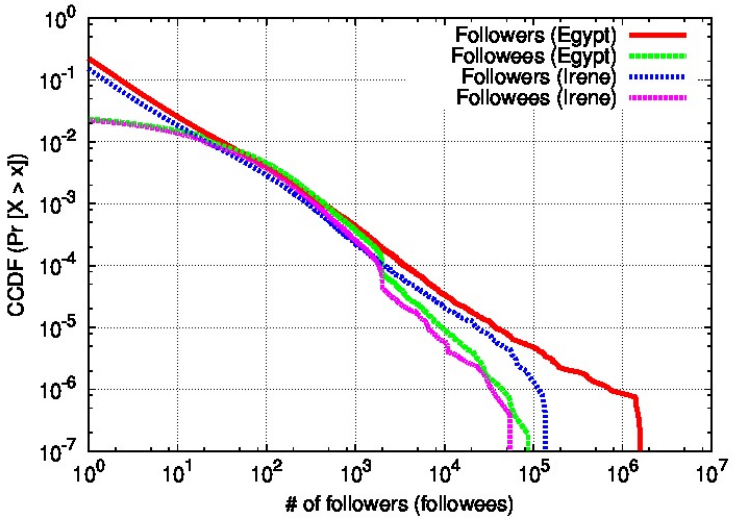
- *Other:* An expression of one's personal feeling, experiences, or sentiments. Remarks that cannot be corroborated. Unrelated random text and less meaningful tweets.

Results are shown for both the Egypt dataset and the Irene dataset in Figure 5.3. Apollo was run with each of the admission control options, and used to return a ranking of top tweets per day in each case. For Egypt dataset, the top 5 tweets per day were selected for grading resulting in a total of 90 tweets graded per experiment (i.e., per admission control option). For Irene dataset, we choose 150 tweets (top 5 tweets per hour for 30 hours). We built a web interface where volunteers could grade these tweets without revealing which tweets were selected in which experiment (i.e., with which admission controller). Once tweets were graded, a *quality score* for each experiment results was computed as the fraction of tweets that have been identified as *fact*. If more than one volunteer graded the same results and differed in classifying a certain tweet, we used the average score.

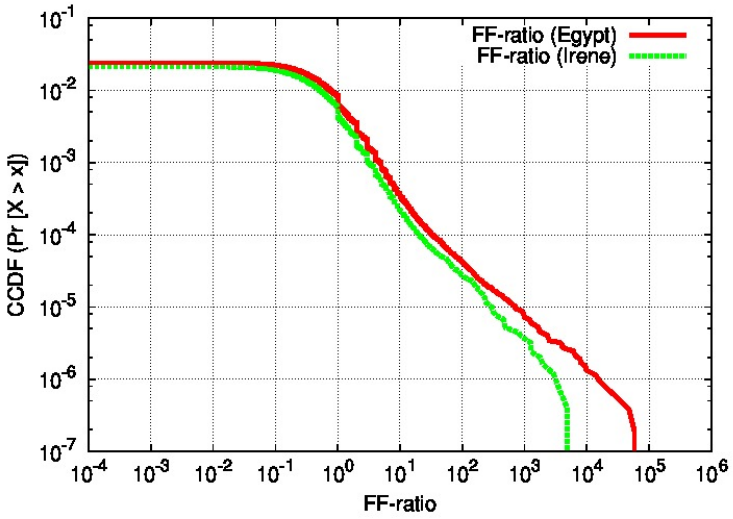
Figure 5.3 presents the *relative* quality scores of various admission control schemes with respect to "no admission control" scheme. In plots, "Beta 1.0" stands for β-controller with threshold, $\tau = 1.0$. We observe that in general β-controllers result in better quality scores. That establishes our hypothesis that diversifying sources does indeed improve the quality of information distillation. We present results with two tweeting options, i) with retweets

and ii) without retweets. The latter option discards all tweets that are explicitly tagged by their sources as "retweets" (i.e., a repeat of tweets posted earlier). We observe that, in both datasets, experiments with no-retweet option produce higher quality scores. This is because they eliminate multiple reporting of the same tweets leaving mostly the tweets experienced or encountered by the respective individuals. For Egypt dataset, simple admission heuristics such as 'no follower', 'no common followee' and 'no descendant' schemes have slightly lower quality scores compared to no admission scheme, but for Irene dataset, they produce considerably higher scores (for no-retweets). Since Irene dataset has limited connectivity, β-controllers could not perform well for it but simply reproduced the base results for with-retweets option, and slightly improved results for no-retweets option.

Figure 5.4 and Figure 5.5 show the percentage of sources and tweets that each admission controller admits for two datasets. It is apparent that some admission schemes are more pessimistic in the sense that they admit fewer sources (and tweets thereby) than others. For Egypt dataset, on an average 15–20% tweets have been pruned by the admission controllers. For Irene dataset, however, admission rates across various admission controllers are considerably high because of the disconnected nature of the underlying social network. Even though the amount of pruning is very small in numbers for Irene dataset, the triage eventually has significant improvements in quality scores (see Figure 5.3(b)).

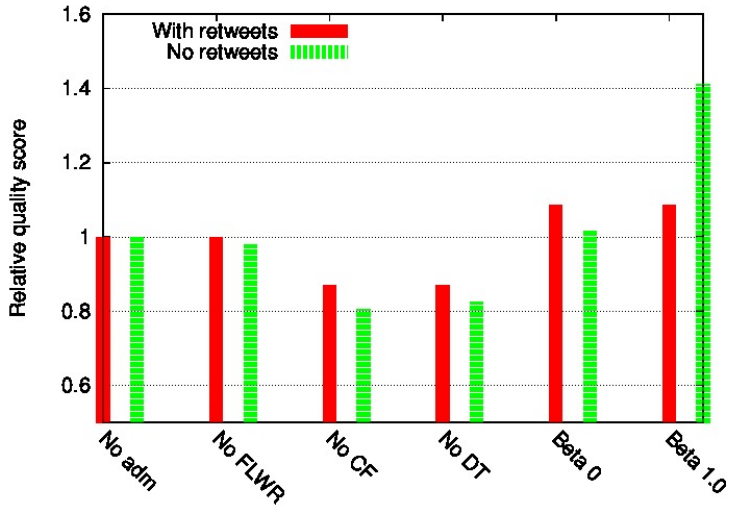


(a)

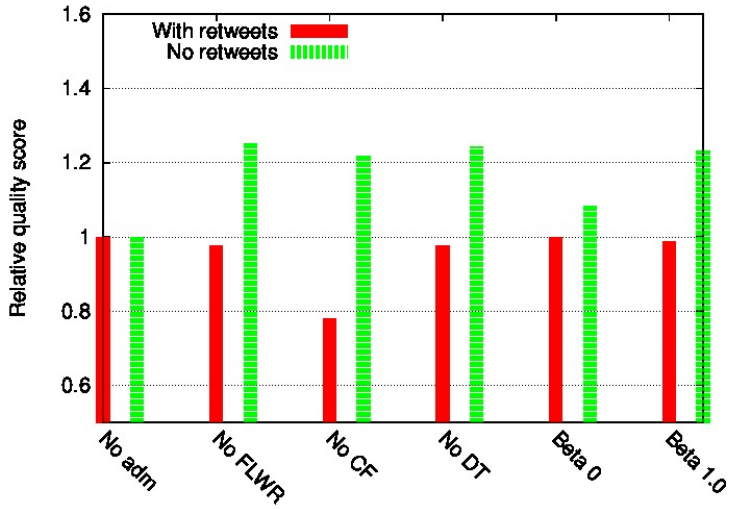


(b)

Figure 5.2: (a) Complementary distribution (CCDF) of follower and followee count per user, (b) CCDF of ff-ratio per user, in Egypt dataset.

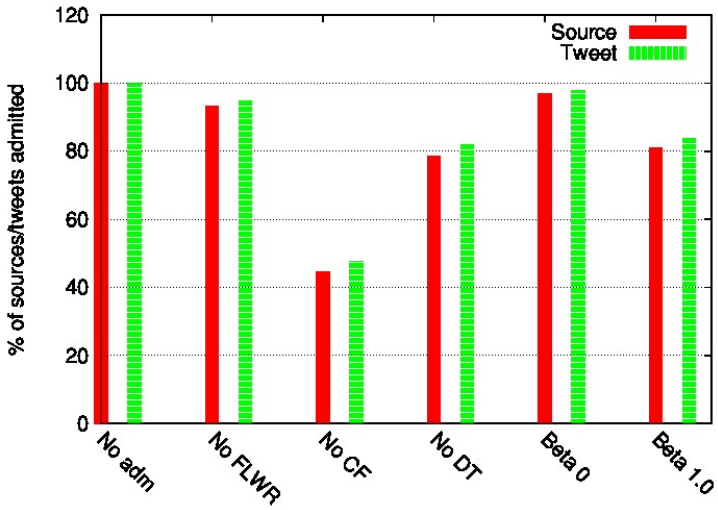


(a) Egypt dataset

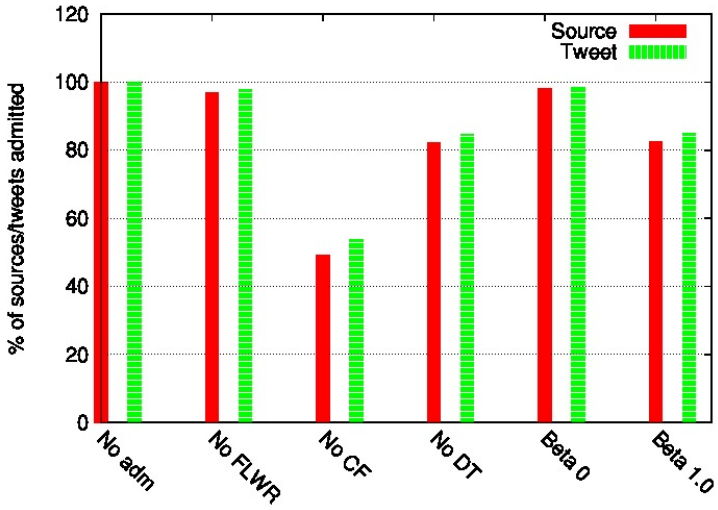


(b) Irene dataset

Figure 5.3: Relative quality scores across different admission control schemes.

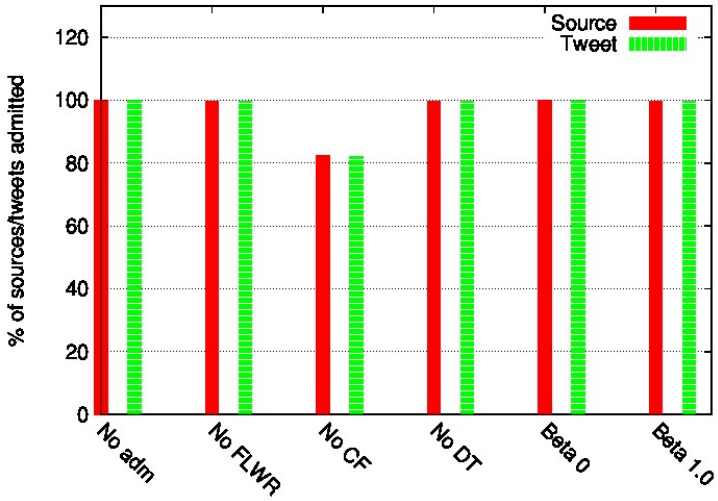


(a) With retweets

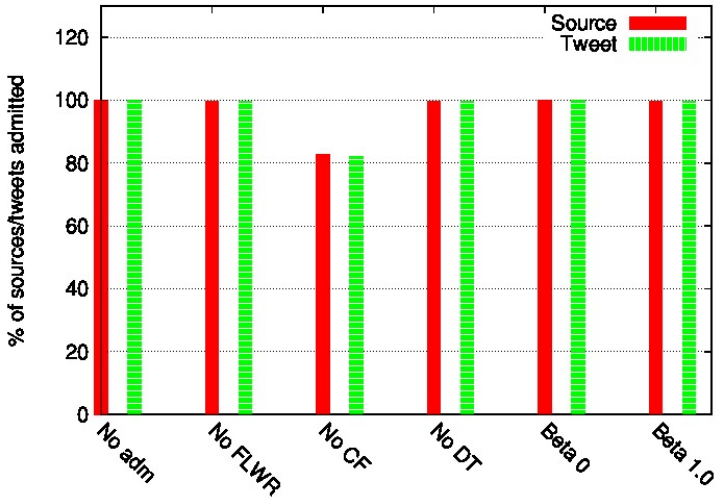


(b) Without retweets

Figure 5.4: Admission controller statistics for different admission schemes (Egypt dataset).



(a) With retweets



(b) Without retweets

Figure 5.5: Admission controller statistics for different admission schemes (Irene dataset).

5.5 Conclusion

We demonstrated that diversifying the sources can improve the results of extracting high quality information (i.e., facts or credible claims) from human-generated content. Human sources on social networks may describe events that do not constitute their own independent observations. This lack of independent corroboration may affect the process of extracting useful information in human generated content. We shows that by considering sources who have more chances of being independent, such adverse effects can significantly reduced.

We considered a fact extraction problem from a large collection of user-generated tweets during two recent events, namely the Egypt unrest and hurricane Irene. We built different online admission controllers that filter tweets based on their sources and feed them into the fact-finding engine, Apollo. We observed that by admitting tweets from a set of more "diverse" sources the fact-finding results significantly were significantly improved. In the current implementation, as a proof-of-concept, we leveraged the "follow" relationship between online users in twitter as an indication of dependence between them. Other attributes that might potentially make sources dependent, such as geographic locations or communities to which users belong, could be investigated in the future.

CHAPTER 6

MAKING APOLLO REAL-TIME AND DISTRIBUTED

6.1 Toward Online Real-Time Social Data Distillation

In the basic version of Apollo, we presented data distillation service when the whole network of sources and claims are given at once. In the first section of this chapter, we investigate the problem of online real-time social data distillation when the network of sources and claims are gradually revealed over time as the sensing activities progress. The result of data distillation also need to accomplish as soon as the next part of the data arrive so that it can handle continuous streams of social data.

6.1.1 Design Approach

As a baseline attempt, we divide the stream of data into chunks. Depending on the purpose of the application, the size of each chunk (measured by time) varies. Each chunk of data is considered as a complete dataset. The data chunk then be fed into the data distillation engine and output shall be produced for the next component in the data processing pipeline. This approach is called *Small Batch Mode*

We then proposed the *Real-time Mode*, in which, credibility of each claim will be assessed as soon as it arrives. Credibility of sources making that claim will be also updated according to an updating rule. Via this updating fashion, credibility estimation of sources shall be improved over time.

Based on these simple designs, we build an application which allows users to observe important events in real-time. The application allows users to create queries which identify events of interest. The application then continously collects data from the social stream provided by Twitter and presents distillation result in real-time with different time window sizes for different levels of sumarization. Even with this simple approach, as we have observed, the application shows considerable promise in providing users with objective report for on-going events.

In this real-time application, data is passed from components to components via Linux pipes. Buffers and queues are implemented simply by temporaty directories in standard Linux file-system. Each query comprises a set of keywords and/or a geo fence. Tweets having specified keyword or within the geo fence shall be harvested from the data stream. Figure 6.1 shows example of a query created via the user interface of the application.

Once a query is provided, a task will be created. A task is an abstraction of a real-time event which being monitored. The task includes all processes, metadata, and data for the event. The status of a task is stored in a text file in JSON format. There is a central background process continously checking and updating statuses of tasks. Task is manipulated via editing the task description file. Figure 6.2 shows how user can interact with tasks. User can pause, resume, or delete a task. User can also export all collected data for more sophisticated and expensive offline analysises.

6.1.2 Implementation

Small Batch Mode In our first implementation attempt, tweets are collected by chunk with different sizes (particularly in 5, 30, and 60 minutes). The most important claims of each chunk shall be presented. Figure 6.3 shows how monitoring result of an event is presented to end users.

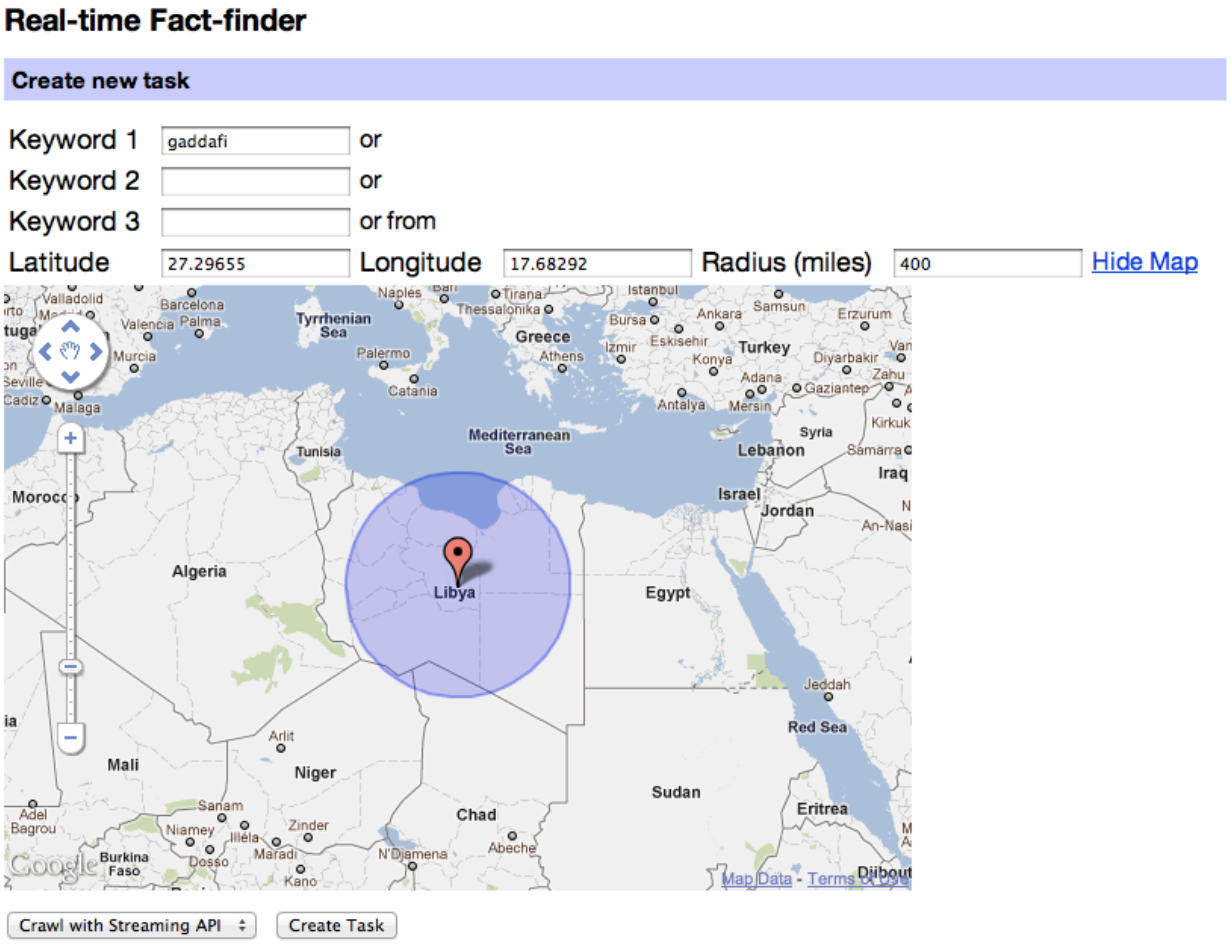


Figure 6.1: The User Interface to create query for real-time event monitoring.

Real-time Mode So far, we have assumed the network of sources and claims is available at once before being fed into the distillation engine. This assumption holds when all data has been gathered and users of the application can wait for the result to be processed. However, there are applications in which having result available in a timely fashion is critical. In such cases, we propose following extension for the distillation engine which processes data as it arrives, and does not need the whole network of sources and claims for assessment. The basic idea behind this extension is that Apollo will maintain a database of credibility of sources, and as new claims arrive, Apollo uses current knowledge about sources credibility to assess claims, and also updates source credibility based on the newly observed claims.

Current tasks

Task ID	Created Time (Central Time)	Running time (Seconds)	Collected Data (Bytes)	Query	Crawler	Status	Actions
1320265082	Wed Nov 2 15:18:02 2011	8384083	1239851008	"#occupy","occupywallstreet","occupywallst",@(-74.00176, 40.71952, 2)	search_api	paused	Delete|Pause|Resume|Export
1320295072	Wed Nov 2 23:37:52 2011	8354094	10715136	"yemen","taiz","ta'izz",@(43.97321, 13.54233, 70)	search_api	paused	Delete|Pause|Resume|Export
1320295448	Wed Nov 2 23:44:08 2011	8353718	233472	"yemen","taiz","",@(43.97321, 13.54233, 70)	search_api	paused	Delete|Pause|Resume|Export
1320372405	Thu Nov 3 21:06:45 2011	8276761	8716288	"yemen","taiz","ta'izz",@(44.05011, 13.58505, 70)	streaming_api	paused	Delete|Pause|Resume|Export
1320411299	Fri Nov 4 07:54:59 2011	8237867	375095296	"#occupy","#occupywalltreet","#occupywallst",@(-88.22528, 40.11415, 1)	streaming_api	paused	Delete|Pause|Resume|Export
1320417402	Fri Nov 4 09:36:42 2011	8231763	1540096	"","","",@(66.28644, 33.65389, 350)	search_api	paused	Delete|Pause|Resume|Export
1320421691	Fri Nov 4 10:48:11 2011	8227475	4668592128	"","","",@(37.50226, 55.74129, 100)	search_api	paused	Delete|Pause|Resume|Export
1320427719	Fri Nov 4 12:28:39 2011	8221447	206204928	"gaddafi","","",@(17.24347, 26.66997, 300)	search_api	paused	Delete|Pause|Resume|Export
1327010456	Thu Jan 19 16:00:56 2012	1638709	368783360	"india rain","","",@(78.23956, 21.70126, 500)	search_api	active	Delete|Pause|Resume|Export
1328390785	Sat Feb 4 15:26:25 2012	258381	7868416	"QLD flood","","",@(-88.22528, 40.11415, 0.0)	search_api	active	Delete|Pause|Resume|Export
1328390846	Sat Feb 4 15:27:26 2012	258320	115634176	"bahrain","","",@(-88.22528, 40.11415, 0.0)	search_api	active	Delete|Pause|Resume|Export
1328390870	Sat Feb 4 15:27:50 2012	258296	33939456	"portsaid","ahly","",@(-88.22528, 40.11415, 0.0)	search_api	active	Delete|Pause|Resume|Export

Figure 6.2: The User Interface to manage real-time event monitoring tasks.

The extension is presented as following.

As claim C_j arrives at time t, we estimate

$$C_j = 1.0 - \prod_{i=1}^{M} (1.0 - A[i,j]S_i^{t-1}) \tag{6.1}$$

Then we update credibility of sources as following

$$S_i^t = \gamma^t \alpha S_i^{t-1} + \gamma^t (1-\alpha) C_j A[i,j] \tag{6.2}$$

In which, C_j represents credibility score of claim j, S_i^t represents latest credibility score of source i at up time t. $A[i,j]$ is equal to 1 if source i makes claim j and equal to 0 otherwise. Initially, S_i^0 is assigned to a prior probability that a claim made by an arbitrary source is true. γ^t is a normalized factor at time t to make sure that credibility of all sources agree to an average credibility score.

Figure 6.3: Example from a real-time event monitoring task.

6.1.3 Evaluation

We evaluate the real-time extension on two application: geo-tagging and time-line recovery.

- *Geo-tagging:* In this "toy" example, we represent the general category of applications where participants search for occurrences of observations of interest and upload to a central server the location tags of such observations. Detecting invasive species in the "What's Invasive" campaign [30] is an instance of such applications.

- *Human Sensors:* In this application, humans act as the sensors, simply tweeting about what they see. Apollo identifies what it believes to be the most credible tweets. We show that Apollo can identify important events as they occur and matches well events reported by the media.

Geo-tagging In our simulation, participants have a certain probability P_r to continue walking and probability $1 - P_r$ to stop and return at each forking of the trail. Participants

who decide to continue have equal odds to select the left and right path. We assume that there is a litter cleaning service run by park that will periodically clean all the litters found in the park. Hence the reported litter locations vary over time. In the experiment, we choose a binary tree with a depth of 4 as the trail map of the park. 50 participants are visiting the park. The pollution ratio of the park is defined as the number of litter locations to the total number of locations in the park. It is set to 0.1. The probability people continue to walk past a fork of a path is set to 95%.

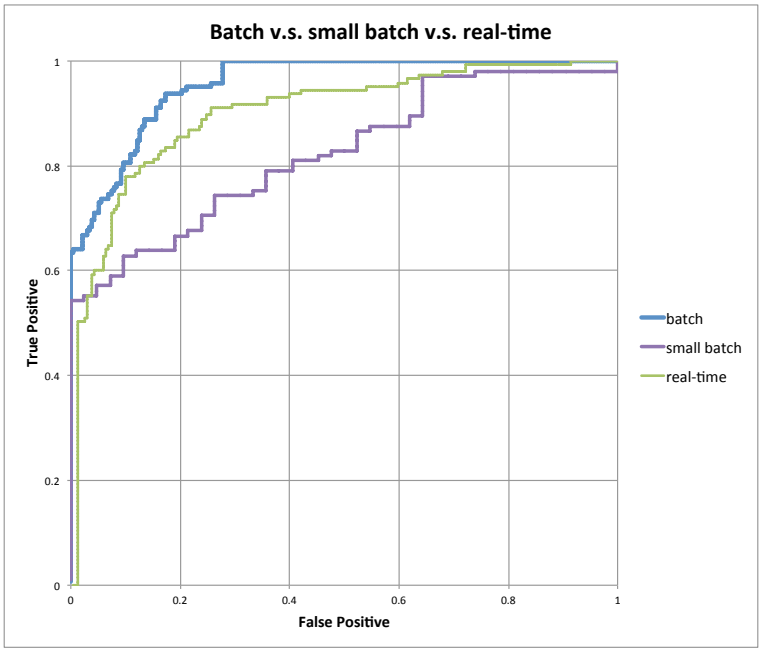


Figure 6.4: Comparing batch distillation, small window batch distillation, and real-time distillation of geo-tagging application.

We compare the batch mode with all claims are provided at once, small batch mode (distillation of small chunks of data from each time windows), and the real-time extesion mode in which claims are assessed as they arrive and credibility of sources is updated as time progresses. The result shows that the accuracy and recall of real-time mode even though are not as good as in batch mode, the tradeoff performance is acceptable; and the real-time mode outperforms small batch mode. Figure 6.4 shows the ROC for performance comparison.

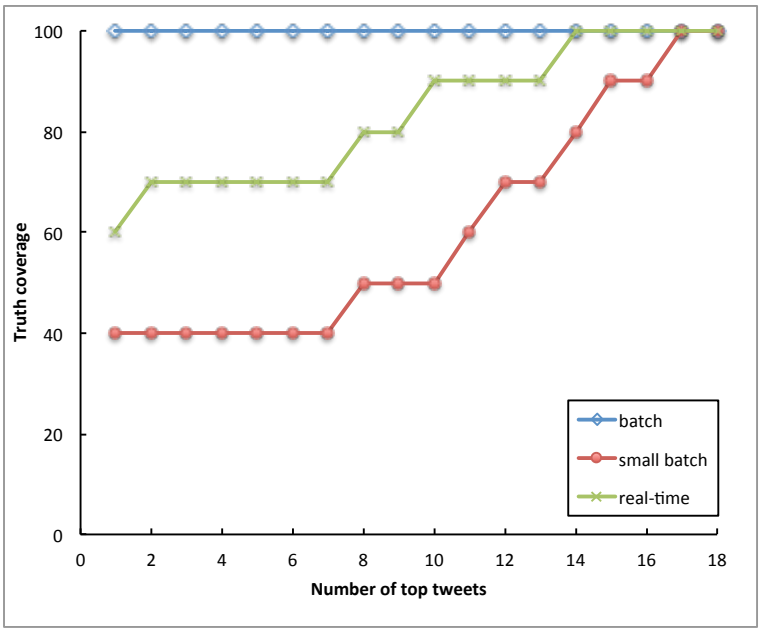


Figure 6.5: Comparing real-time mode with batch and small batch mode

Time-line recovery *Real-time mode extension:*

We evaluate the real-time mode extension in this application. The result shows that real-time mode outperforms the small batch mode (when the window is small enough). Figure 6.5 shows the detailed results in the Egypt dataset. On a side note, the compuation time of real-time mode is significantly lower than the small batch mode. In the Egypt dataset case, on the same hardware configuration, it takes on average about 26 minutes to process data of each hour in small batch mode, while it takes on average under 8 minutes in real-time mode.

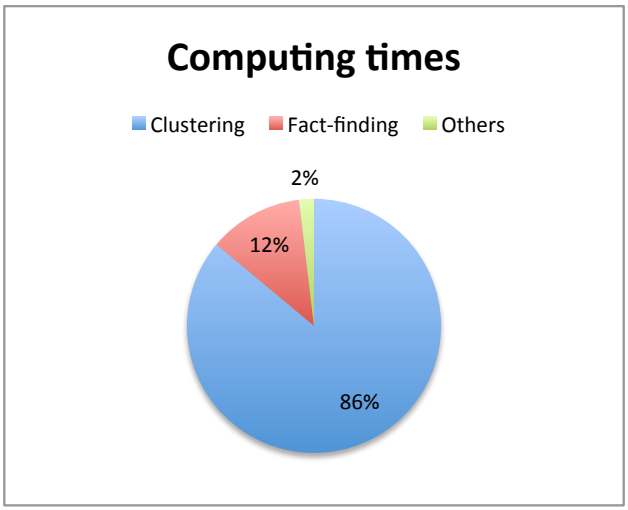


Figure 6.6: Computing Distribution Time of Centralized Apollo

6.2 Toward A Scalable and Distributed System for Social Data Distillation

The current centralized implementation of Apollo has following bottlenecks: (i) Clustering claims is done in one process only, which makes it problematic when the stream of data has more claims than a process can handle in the chosen time window; and (ii) EM style calculation for credibility of sources and claims also done via a single process. Figure 6.6 show computing time distribution of Apollo in different stages in centralized implemenation. As we can see, clustering and fact-finding time dominate the whole distillation process.

Based on above observation, we focus on making the clustering and fact-finding process scalable and distributed.

6.2.1 Design and Implementation Approach

To make Apollo distributed, data and computing result need to be moved around a cluster of computers effectively. Among different options, we chose ZeroMQ [34] for its simplicity and proven performance.

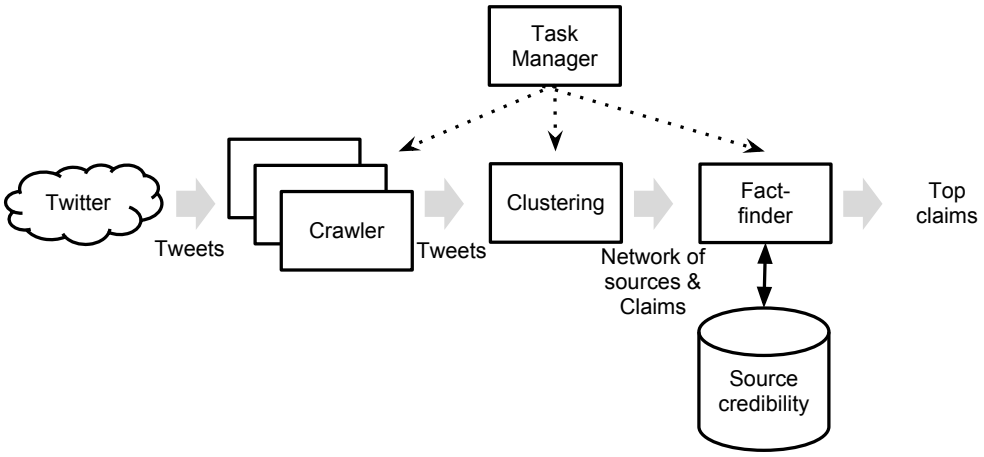


Figure 6.7: Workflow of an Apollo task

As in centralized Apollo, queries are handled by a computing abstraction called task. In distributed Apollo, a task has a configuration file (see Appendix B for an example of such a file) and the task runtime engine will use that configuration file to create and distribute computation accross machines in the cluster.

All processes of Apollo real-time tasks are controled by a background process called $TaskManager$. $TaskManager$ takes a task configuration file as a "recipe" and create processes with appropriate parametters for the task. $TaskManager$ returns an URL for each task, with which users can observe sumarization of the event in real-time from a web brower. The overview of workflow of Apollo task is shown in Figure 6.7.

Scaling up Clustering Process Clustering has been always the most significant bottleneck of Apollo so far. There is extensive bodies of work on clustering so far [35]. Among the known algorithms, we chose Canopy Clustering (for its scalability and ease of distributed implementation) as part of built-in algorithm for distributed Apollo. More information about Canopy Clustering can be found here [36]. We integrate this algorithm into Apollo from a base implementation from Mahout [37]. The integrated system works as following.

As each chunk of data need to be clustered available, we consider the size of the collection.

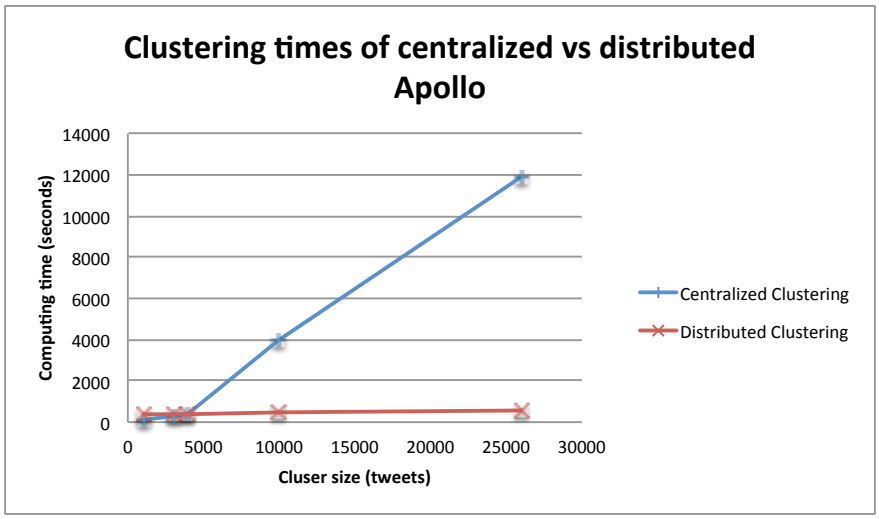


Figure 6.8: Clustering time comparison of centralized Apollo vs distributed Apollo

As it will be shown, the fixed overhead of the distributed clustering implementation is not neglegible. In our experiment cluster, it took about 5 minutes for the distributed clustering component to cluster a minimal set of claims. The overhead is mainly due to setting up and tearing down computing workers and shared data repository on different machines. Due to this, if the size of the claim set is smaller than a threshold, the centralized clustering algorithm is employed. Otherwise, the claim set will be sent to a Mahout cluster and appropriate clustering job shall be created by Apollo. The clustering result then be interpreted back to Apollo format and the normal distillation process is continued. The computing time comparision of the centralized and distributed clustering is shown in Figure 6.8. As we can see, if the claim set is smaller than 4000 items, centralized implementation takes less time and the distributed implementation. We chose this as the trigger point to send data for distributed clustering component.

In our experimental implementation, the distributed clustering run in a cluster of 10 servers, each with 4 computing cores, 16G RAM, 2TB RAID-1 hard-drive, the servers in this cluster are connected by dual Gigabit links. The centralized clustering algorithm is run on a single server with 6 computing cores, 32G RAM, 2TB RAID-1 hard-drive.

Scaling up Fact-finding Process Apollo was designed with a goal that each component (crawling, clustering, fact-finding, result representing, etc.) only need to agree to fixed interfaces (which are reprented by the formats of the data exchanged among components); beyond that, the components are not restricted to any particular implementation requirement. In centralized Apollo, the fact-finding component has three different implementations for three different fact-finding algorithms from as simple as voting, to as complicated as EM-style fact-finding. To scale up the fact-finding process, we need to provide Apollo with a scalable implementation option. Fortunately, as the result represented in the real-time Apollo, the real-time extension of Apollo show encouraging performance advantage.

We improve the real-time extension by storing crediblity of sources in a distributed database (we use HBase in this case), a collection of fact-finding processes can handle multiple claims as they arrive and then update credibility of relevant sources accordingly. We assume that each source shall not produce a large number of claims simultaneously, thus, the requirement on real-time consistency of an off-the-shell distributed database like HBase is sufficient. The performance result shows that the distributed fact-finding component designed as described gives a performance boost propotionally to the number of distributed fact-finding worker processes.

6.3 Conclusion and Future Work

We also presented the design approaches, experimental implementations, and initial evaluation results for two extensions of Apollo to make it able to handle real-time data streams, as well as able to be deployed in a distributed manner for better scalability. The results show promise for future production-grade implementation. We would like to leave a more sophisticated distributed fact-finding algorithm as part of future works of Apollo.

CHAPTER 7

RELATED WORK

7.1 Communication Layer

7.1.1 A Control Theory Approach to Throughput Optimization in Multi-Channel Collection Sensor Networks

Data collection from static nodes is one of the most important problems in sensor networks. Examples of data collection networks include habitat monitoring [38], structural monitoring [39], countersniper protection [40], and applications of TinyDB [41]. The importance of this network model leads to several network protocols that are optimized for data collection applications. For example directed diffusion [42] allows sinks to declare interests that facilitate data collection from the network.

This dissertation presents a protocol for load balancing across channels for throughput maximization. The idea of using multiple channels in wireless networks is not new. One example of a multi-channel MAC-layer protocol is MMAC [6], proposed for wireless communication using 802.11. However, as clearly articulated in [43], protocols used for 802.11 nodes are too heavy-weight for sensor networks. There were also schemes based on frequency hopping [44, 45]. These schemes all require nodes to switch among channels frequently even when there is no need for transmission. With current technology, these schemes are not well-suited for wireless sensor network devices in which the time to switch channels is non-negligible and the power consumption for switching is a major disadvantage. Furthermore, they also require synchronization among nodes which makes them even harder to apply to

wireless sensor networks. There exists other work on exploiting frequency diversity based on devices with multiple radio interfaces [46, 7] or with interfaces which can listen simultaneously on different channels [5, 4]. We have not observed any hardware platforms for wireless sensor networks that support these. In the sensor network domain, only a few papers have evaluated the use of multiple channels in communication [8, 47]. Unfortunately, they did so in simulation.

Our study is the first implementation and empirical evaluation of a sensor network-based data collection protocol that utilizes multiple channels for improving throughput and uses feedback control to adaptively allocate nodes to channels for best use of channel diversity. A novel aspect of our protocol is the use of control theory for adapting channel allocation.

7.1.2 A Practical Multi-Channel Media Access Control Protocol for Wireless Sensor Networks

The idea of multi-channel MAC protocols is not new in the wireless network research community. In the ad-hoc network domain, there have already been some practical systems utilizing multiple channels for communication. However, most of them have a very simple network topology, assuming only one base-station with the remaining nodes communicating with the base-station within only one hop. Others use heavy-weight protocols which are not suitable for sensor network devices [43].

There has also been work based on channel hopping [44, 45]. These schemes require synchronization and frequent switching among channels even when there is no need for communication. Unless supported by the physical layer, switching channels in sensor devices costs a great deal of time and energy which makes these schemes unattractive for WSNs. In our experiments with MicaZ motes [48], the time to switch between two channels and wait until the frequency synthesizer stabilizes is roughly equal to the time to transmit one packet. Therefore, nodes which frequently switch channels run out of battery faster than

other nodes. Although the radio technology is advancing, it is nontrivial to reduce the cost of channel switching in such strictly constrained devices as sensor nodes. There exist efforts applied for devices with multi-radio interfaces [7, 46], or with specially designed interfaces which can listen simultaneously to different channels [5, 4]. However, as we have observed, hardware equipped with one half-duplex radio interface are much more popular.

There are also efforts in industry which utilize multi-channel radios. TSMP [49] maintains synchronization among nodes. Nodes employ frequency hopping according to a shared pseudo-random schedule. TSMP requires both sychronization and frequent channel switching. There is also ongoing effort on the SP100.11a standard [50]. This standard uses a simplified 3-channel hopping scheme. Sexton and others have documented narrow band fading problems and promote multi-channel communication as added diversity.

Finally, there are results in both ad-hoc multi-hop networks and WSNs focusing on systems with only one half-duplex radio interface [6, 47, 51]. Unfortunately, all of them are done in simulation and most of them assume that the time to switch channels is negligible. There are also protocols that have a working implementation showing significant improvements in total network throughput. However, these protocols are limited to a specific traffic pattern such as data collection and aggregation [9], or data dissemimation [10, 11]. Networks with arbitrary traffic patterns are not yet covered.

7.2 Information Layer

7.2.1 Apollo: A Data Distillation Service Social Sensing

Human-centric sensing and in particular participatory sensing has received significant attention in the past years. Participatory sensing was introduced in [52] and a broad overview of such applications is summarized in [53]. Some of the early participatory sensing applications include CenWits [54], a participatory sensor network to search and rescue hikers, CarTel [55],

a vehicular sensor network for traffic monitoring, and BikeNet [56], a bikers sensor network for monitoring popular bike routes. More recent applications include CabSense [57] to find taxi cabs in New York city and a cooperative transit tracking using GPS enabled smartphones [31]; or [58] collect trip information from a large fleet of taxi to predict expected fare and trip duration. Our service, Apollo, can be leveraged by the variety of participatory sensing applications to distill the sensor data collected and improve their accuracy.

A relevant body of work in the machine learning and data mining communities performs trust analysis based on the source and claim information network. When seeking to determine whether or not something should be believed, the simplest approach is to take a vote, accepting the claim supported by the most information sources in a set of mutually exclusive claims, or those claims supported by a number of sources exceeding a threshold. However, this implicitly assumes that all information sources are equally trustworthy. Fact-finders, a class of iterative trust analysis algorithms, avoid this assumption by seeking both the believability of the claims *and* the trustworthiness of the sources. In each iteration, the trustworthiness of each source is calculated from the believability of the claims it makes, and the believability of each claim is calculated from the trustworthiness of the sources asserting it, repeating until convergence.

Hubs and Authorities [25], for example, can be adapted as a simple fact-finder whether the belief in a claim c is $B(c) = \sum_{s \in S_c} T(s)$ and the trustworthiness of a source s is $T(s) = \sum_{c \in C_s} B(c)$, where S_c and C_s are the sources asserting a given claim and the claims asserted by a particular source, respectively. Other straightforward instances of this model include TruthFinder [59] and the Investment, PooledInvestment and Average·Log algorithms [21]. Many fact-finders also enhance the basic formula. 3-Estimates [20] rewards sources that correctly assert highly disputed claims, while AccuVote [60, 61] considers "source dependence", where one source derives some of its information from one or more other sources, effectively boosting the trustworthiness of independent sources. The problem of detecting

source dependency has also been discussed and solutions have been proposed [62, 63]. Blanco et al. [64] analyze the source dependency with a focus on copy detection using knowledge from multiple attributes. Recently, the need for data distillation using credibility is shown in [65]. [65] proposed an application-specific method to estimate reputation of users to enhance large scale participatory acoustic sensor data analysis. Apollo can be considered as a next step toward a general purpose approach of using credibility to enhance data analysis.

Frameworks have been proposed to enhance fact-finding algorithms in general: Pasternack et al. [21] incorporate prior knowledge concerning the claims (in the form of first-order logic) into fact-finding to leverage what the user already knows. A consequent piece of work [66] introduces a broad range of background knowledge into the process. Gupta et al.[67] account for a source's varying expertise across different topics. Additionally, trust analysis has been done both on a homogeneous network of information providers [68] or claims [69] and a heterogeneous network consisting of multi-typed objects [70]. We proposed a maximum likelihood estimator that offers a joint optimal (in the maximum likelihood sense) estimation on source reliability and claim correctness based on a group of general simplifying assumptions [71]. In following work, we further extended their model to incorporate the background bias of a randomly chosen claim to be true and quantified the accuracy of the maximum likelihood estimation of source reliability [72]. Apollo borrows the idea of fact-finding from the basic iterative model. In order to support as many different sensing applications as possible, we avoid using methods that rely on source-dependence or prior knowledge.

7.2.2 Source Selection in Social Sensing Applications

Social sensing has received much attention in recent years [73]. This is due to the large proliferation of devices with sensing and communication capabilities in the possession of average individuals, as well as the availability of ubiquitous and real-time data sharing

opportunities via mobile phones with network connection and via social networking sites (i.e., Twitter). A few early applications include CarTel [74], a vehicular data collection and sharing system, BikeNet [75], bikers sharing their biking experiences in different trails, PhotoNet [76], volunteers collecting pictures from a disaster scene, CenWits [77], search and rescue scheme for hikers, CabSense [57], participatory sensing application with taxi car fleets, and ImageScape [78], an application for sharing diet experiences.

Social sensing involves humans as the active data generator. Here, the human acts as a sensor. One problem with social sensing is the abundance of noisy data, as humans are not as reliable as well-calibrated sensors. A significant amount of efforts have been made by researchers to extract useful information from a vast pool of such noisy data. For example, following techniques inspired by generalizations of Google's PageRank [26], information can be represented by a source-claim network [25, 79, 80] that simply tells who said what. An iterative algorithm then tries to reason on this graph to extract out the most trustworthy information given the degree of corroboration and the inferred source reliability. Generally these techniques are called *fact-finders*, a class of iterative algorithms that jointly infer credibility of claims as well as trustworthiness of sources. Hubs and Authorities [25] is a simple fact-finder where belief in correctness of a claim is computed as the sum of trustworthiness of sources who made that claim, and the trustworthiness of a source is in turn obtained from the beliefs in correctness of the claims it makes. Notable fact-finding schemes also include TruthFinder [27], 3-Estimates [81], and AccuVote [82, 83].

There are several extensions developed to improve fact-finding results, such as incorporating prior knowledge [21, 66], and accounting for the source's expertise in different topics [84]. Most recently, a maximum likelihood estimation approach was developed that is the first to compute an *optimal solution* to the credibility assessment problem [33]. The solution is optimal in the sense that the resulting assignment of correctness values to claims and sources is the one of maximum likelihood. Using results on error bounds of maximum-likelihhood

estimators, a confidence interval is also computed in the maximum-likelihood hypothesis [85].

The problem of information source selection has been discussed in data retrieval and web context [86, 87, 88] and in query sampling [89, 90, 91]. These works mainly reason on attributes of sources as well as the content that those sources generate. In contrary, ours is a content-agnostic approach that rely only on relationship among sources.

In this study, we use Apollo [32] as plug-ins for a versatile set of applications. We use the aforementioned maximum-likelihood estimator [33] as the fact-finding algorithm in Apollo. The estimator makes the inherent assumption that observations across different sources are independent. Clearly, in practical social sensing applications, such independence may not be really the case. We demonstrate that the performance of fact-finding can be significantly improved by using simple heuristics for diversifying sources so that the information network contains sources that are less dependent on one another.

While diversifying sources would not be needed if one could accurately account for dependence between sources in data credibility assessment, we argue that, in general, estimating the degree of dependence between sources is very hard. For example, if one source follows another on Twitter and both report the same observation, it is hard to tell whether the second report is simply a relay of the first, or is an independent measurement. Given the ambiguity regarding the originality (versus dependence) of observations, we therefore suggest that diversifying the sources is a useful technique whether or not credibility assessment can take dependence into account.

We implemented our resulting source selection scheme as an online admission controller that is included as an upfront plug-in to the Apollo execution pipeline. Results show that our admission control can both speed up data processing (by reducing the amount of data to be processed) and improve credibility estimates (by removing dependent and correlated sources).

CHAPTER 8

CONCLUSION AND FUTURE WORK

We presented research results accross two different layers of data to decision pipelines in embedded and social sensing systems. Starting with solutions that maximize data throughput, and progressing to solutions that enable computing systems to selectively retain and process important pieces of information.

This dissertation confirmed the hypothesis that it is possible to build application-independent data to decision services for embedded and social sensing applications that can collect data, distill different data formats including boolean observations, time-series data, as well as unstructured data such as text and images.

Many interesting aspects of designing a good data distillation service remain candidates for future work. Importantly, we have not investigated the robustness of the service to malicious use. In principle, if the algorithm used to determine the credibility of claims is known, it becomes possible for colluding agents to foil it; a topic we have not addressed. Individuals can also gain credibility by consistently providing true observations, then exploit that accumulated credibility to insert bad data into the mix and have it pass the distillation filter. This problem is common to reputation-based systems. It remains relevant here since the underlying algorithm tries to estimate source credibility as well. We also have not addressed the utilization of prior knowledge in the distillation process [24], which, we believe to be another important extension of Apollo.

This thesis may be the forerunner of a new type of information browsers that collect, distill, and process real-time information streams in response to user queries. With the

rising use of social networking sites like Facebook, Twitter, Google+, and alike, information about the world is produced in real-time in an unprecedented volume and detail. This gives a unique opportunity for building computing systems that handle real-time data and respond to queries about current state at different parts of the world. The possibility exists for most system builders with the openness of social network sites like Twitter, and Google+, in which users are encouraged to let their streams of updates publicly accessible.

In events like the Egypt Unrest and Occupy Wall Street, it was clear that social networking sites become a main mechanism for millions to collaborate, report, and react, towards a common perception about what was really happening. During these events, millions of updates were created by thousand or millions of people. It is important for those updates to be processed, ranked, and distilled so that humans can objectively know what is happening and make their own judgement.

Distilling a stream of real-time updates from social networks requires a fresh new look at how a "search engine" should be built on top of the data stream to give the most valuable information to users. This study is an initial step towards such a search engine.

APPENDIX A

DETAILED PROOF FOR BAYESIAN APPROACH

Consider an assertion C_j made be several sources $S_{i_1}, ..., S_{i_K}$. Let $S_{i_k}C_j$ denote the fact that source S_{i_k} made assertion C_j. We further assume that Equation (4.5) and Equation (4.6) hold. In other words:

$$\frac{P(S_{i_k}C_j|C_j^t)}{P(S_{i_k}C_j)} = 1 + \delta_{i_kj}^t$$
$$\frac{P(S_{i_k}C_j|C_j^f)}{P(S_{i_k}C_j)} = 1 + \delta_{i_kj}^f$$

where $|\delta_{i_kj}^t| << 1$ and $|\delta_{i_kj}^f| << 1$.

Under these assumptions, we prove that the joint probability $P(S_{i_1}C_j, S_{i_2}C_j, ..., S_{i_K}C_j)$, denoted for simplicity by $P(Sources_j)$, is equal to the product of marginal probabilities $P(S_{i_1}C_j), ..., P(S_{i_K}C_j)$.

First, note that, by definition:

$$\begin{aligned} P(Sources_j) &= P(S_{i_1}C_j, S_{i_2}C_j, ..., S_{i_K}C_j) \\ &= P(S_{i_1}C_j, S_{i_2}C_j, ..., S_{i_K}C_j|C_j^t)P(C_j^t) \\ &+ P(S_{i_1}C_j, S_{i_2}C_j, ..., S_{i_K}C_j|C_j^f)P(C_j^f) \end{aligned} \tag{A.1}$$

Using the conditional independence assumption, we get:

$$\begin{aligned} P(Sources_j) &= P(C_j^t)\prod_{k=1}^{K} P(S_{i_k}C_j|C_j^t) \\ &+ P(C_j^f)\prod_{k=1}^{K} P(S_{i_k}C_j|C_j^f) \end{aligned} \tag{A.2}$$

Using Equation (4.5) and Equation (4.6), the above can be rewritten as:

$$\begin{aligned} P(Sources_j) &= P(C_j^t)\prod_{k=1}^{K_j}(1+\delta_{i_kj}^t)\prod_{k=1}^{K_j} P(S_{i_k}C_j) \\ &+ P(C_j^f)\prod_{k=1}^{K_j}(1+\delta_{i_kj}^f)\prod_{k=1}^{K_j} P(S_{i_k}C_j) \end{aligned} \tag{A.3}$$

and since $|\delta_{i_kj}^t| << 1$ and $|\delta_{i_kj}^f| << 1$, any higher-order terms involving them can be ignored. Hence, $\prod_{k=1}^{K_j}(1+\delta_{i_kj}^t) = 1 + \sum_{k=1}^{K_j}\delta_{i_kj}^t$, which results in:

$$\begin{aligned} P(Sources_j) &= P(C_j^t)(1+\sum_{k=1}^{K_j}\delta_{i_kj}^t)\prod_{k=1}^{K} P(S_{i_k}C_j) \\ &+ P(C_j^f)(1+\sum_{k=1}^{K_j}\delta_{i_kj}^f)\prod_{k=1}^{K} P(S_{i_k}C_j) \end{aligned} \tag{A.4}$$

Distributing multiplication over addition in Equation (A.4), then using the fact that $P(C_j^t)+P(C_j^f) = 1$ and rearranging, we get:

$$P(Sources_j) = \prod_{k=1}^{K_j} P(S_{i_k}C_j)(1+Terms_j) \tag{A.5}$$

where:

$$Terms_j = P(C_j^t) \sum_{k=1}^{K_j} \delta_{i_k j}^t + P(C_j^f) \sum_{k=1}^{K_j} \delta_{i_k j}^f \tag{A.6}$$

Next, it remains to compute $Terms_j$.

Consider $\delta_{i_k j}^t$ as defined in Equation (4.5). We can rewrite the equation as follows:

$$\delta_{i_k j}^t = \frac{P(S_{i_k} C_j | C_j^t) - P(S_{i_k} C_j)}{P(S_{i_k} C_j)} \tag{A.7}$$

where by definition, $P(S_{i_k} C_j) = P(S_{i_k} C_j | C_j^t) P(C_j^t) + P(S_{i_k} C_j | C_j^f) P(C_j^f)$. Substituting in Equation (A.7), we get:

$$\begin{aligned} \delta_{i_k j}^t &= \frac{P(S_{i_k} C_j | C_j^t)(1 - P(C_j^t)) - P(S_{i_k} C_j | C_j^f) P(C_j^f)}{P(S_{i_k} C_j | C_j^t) P(C_j^t) + P(S_{i_k} C_j | C_j^f) P(C_j^f)} \end{aligned} \tag{A.8}$$

Using the fact that $1 - P(C_j^t) = P(C_j^f)$ in the numerator, and rearranging, we get:

$$\delta_{i_k j}^t = \frac{(P(S_{i_k} C_j | C_j^t) - P(S_{i_k} C_j | C_j^f)) P(C_j^f)}{P(S_{i_k} C_j | C_j^t) P(C_j^t) + P(S_{i_k} C_j | C_j^f) P(C_j^f)} \tag{A.9}$$

We can similarly show that:

$$\begin{aligned} \delta_{i_k j}^f &= \frac{P(S_{i_k} C_j | C_j^f) - P(S_{i_k} C_j)}{P(S_{i_k} C_j)} \\ &= \frac{P(S_{i_k} C_j | C_j^f)(1 - P(C_j^f)) - P(S_{i_k} C_j | C_j^t) P(C_j^t)}{P(S_{i_k} C_j | C_j^t) P(C_j^t) + P(S_{i_k} C_j | C_j^f) P(C_j^f)} \\ &= \frac{(P(S_{i_k} C_j | C_j^f) - P(S_{i_k} C_j | C_j^t)) P(C_j^t)}{P(S_{i_k} C_j | C_j^t) P(C_j^t) + P(S_{i_k} C_j | C_j^f) P(C_j^f)} \end{aligned} \tag{A.10}$$

Dividing Equation (A.9) by Equation (A.10), we get:

$$\frac{\delta^t_{i_k j}}{\delta^f_{i_k j}} = -\frac{P(C^f_j)}{P(C^t_j)} \tag{A.11}$$

Substituting for $\delta^t_{i_k j}$ from Equation (A.11) into Equation (A.6), we get $Terms_j = 0$. Substituting with this result in Equation (A.5), we get:

$$P(Sources_j) = \prod_{k=1}^{K_j} P(S_{i_k} C_j) \tag{A.12}$$

The above result completes the proof. We have shown that the joint probability $P(S_{i_1} C_j, S_{i_2} C_j, ..., S_{i_K} C_j)$, denoted for simplicity by $P(Sources_j)$, is well approximated by the product of marginal probabilities $P(S_{i_1} C_j), ..., P(S_{i_K} C_j)$. Note that, the proof did not assume independence of the marginals. Instead, it proved the result under the small $\delta_{i_k j}$ assumption.

APPENDIX B

TASK CONFIGURATION FILE EXAMPLE

Listing B.1: Example of A Task Configuration File for Distributed Apollo

```
{
  "type": "fact-finder:bayesian",
  "id": "test-bayesian",
  "plugins": [
    {
      "type": "data_sources:twitter_file_stream",
      "id": "p1",
      "out_pipe": {
        "host": "127.0.0.1",
        "port": 5555,
        "out_types": ["apollo_tweet"]
      },
      "options": {
        "data_file": "/data_sources/twitter_file_stream/tweets.txt"
      }
    },
    {
      "type": "stream_chunker:tweet_chunker",
```

```
  "id": "p2",
  "in_pipe": {
    "host": "127.0.0.1",
    "port": 5555,
    "in_types": ["apollo_tweet"],
    "passing_types": ["apollo_tweet"]
  },
  "out_pipe": {
    "host": "127.0.0.1",
    "port": 6666,
    "out_types": ["apollo_tweet_pair:pair",
                  "apollo_tweet_pair:flush"],
    "passing_types": ["apollo_tweet"]
  },
  "options": {
    "time_window": 3600
  }
},
{
  "type": "distance_functions:tweet_jaccard",
  "id": "p3",
  "in_pipe": {
    "host": "127.0.0.1",
    "port": 6666,
    "in_types": ["apollo_tweet_pair:pair",
                 "apollo_tweet_pair:flush"],
```

```
    "passing_types": ["apollo_tweet", "apollo_tweet_pair:flush"]
  },
  "out_pipe": {
    "host": "127.0.0.1",
    "port": 7777,
    "out_types": ["apollo_tweet_pair:distance"],
    "passing_types": ["apollo_tweet", "apollo_tweet_pair:flush"]
  }
},
{
  "type": "clustering:basic_clustering",
  "id": "p4",
  "in_pipe": {
    "host": "127.0.0.1",
    "port": 7777,
    "in_types": ["apollo_tweet_pair:distance",
                 "apollo_tweet_pair:flush"],
    "passing_types": ["apollo_tweet", "apollo_tweet_pair:flush"]
  },
  "out_pipe": {
    "host": "127.0.0.1",
    "port": 8888,
    "out_types": ["apollo_tweet_cluster:cluster"],
    "passing_types": ["apollo_tweet", "apollo_tweet_pair:flush"]
  },
  "options": {
```

```
    "max_distance": 0.3,
    "max_cluster": 50000
  }
},
{
  "type": "fact_finders:bayesian",
  "id": "p5",
  "in_pipe": {
    "host": "127.0.0.1",
    "port": 8888,
    "in_types": ["apollo_tweet_cluster:cluster", "apollo_tweet",
                 "apollo_tweet_pair:flush"],
    "passing_types": ["apollo_tweet", "apollo_tweet_pair:flush"]
  },
  "out_pipe": {
    "host": "127.0.0.1",
    "port": 9999,
    "out_types": ["apollo_cred:cluster", "apollo_cred:source"],
    "passing_types": ["apollo_tweet", "apollo_tweet_pair:flush",
                      "apollo_tweet_cluster:cluster"]
  }
},
{
  "type": "fact_finder_sink:basic_sink",
  "id": "p6",
  "in_pipe": {
```

```
      "host": "127.0.0.1",
      "port": 9999,
      "in_types": ["apollo_cred:cluster", "apollo_cred:source",
                   "apollo_tweet", "apollo_tweet_pair:flush",
                   "apollo_tweet_cluster:cluster"]
    },
    "options": {
      "output_dir": "/workflows/bayesian/test_output_dir"
    }
  },
  {
    "type": "fact_finder_view:basic_view",
    "id": "p7",
    "options": {
      "input_dir":
      "/workflows/bayesian/test_output_dir"
    }
  }
]
}
```

REFERENCES

[1] J. C. Doyle, B. A. Francis, and A. R. Tannenbaum, *Feedback control theory.* New York: MacMillan, 1992. [Online]. Available: citeseer.ist.psu.edu/doyle90feedback.html

[2] K. Srinivasan and P. Levis, "RSSI is under appreciated," in *Proceedings of The Third Workshop on Embedded Networked Sensors (EmNets 2006)*, 2006.

[3] M. Andersson, D. Henriksson, and A. Cervin, "TrueTime: Simulation of networked and embedded control systems," Home page, http://www.control.lth.se/truetime, 2006.

[4] A. Nasipuri and S. R. Das, "Multichannel csma with signal power-based channel selection for multihop wireless networks," in *Proceedings of IEEE VTC'00*, 2000.

[5] N. Jain, S. R. Das, and A. Nasipuri, "A multichannel csma mac protocol with receiver-based channel selection for multihopwireless networks," in *Proceedings of IEEE IC3N'01*, 2001.

[6] J. So and N. H. Vaidya, "A multi-channel mac protocol for ad hoc wireless networks," in *Proceedings of ACM Mobihoc'04*, 2004.

[7] A. Adya, P. Bahl, J. Padhye, A. Wolman, and L. Zhou, "A multi-radio unification protocol for ieee 802.11 wireless networks," in *Proceedings of IEEE Broadnets'04*, 2004.

[8] G. Zhou, C. Huang, T. Yan, T. He, J. A. Stankovic, and T. F. Abdelzaher, "MMSN: Multi-frequency media access control for wireless sensor networks," in *Proceedings of the IEEE Infocom*, 2006.

[9] H. K. Le, D. Henriksson, and T. Abdelzaher, "A control theory approach to throughput optimization in multichannel collection sensor networks," in *IPSN 2007*, 2007.

[10] R. Simon, L. Huang, E. Farrugia, and S. Setia, "Using multiple communication channels for efficient data dissemination in wireless sensor networks," in *MASS 2005*, 2005.

[11] L. Wang and S. S. Kulkarni, "*appa*: Gossip based multi-channel reprogramming for sensor networks." in *DCOSS*, 2006, pp. 119–134.

[12] P. Levis, "Tep 116: Packet protocols," http://www.tinyos.net/tinyos-2.x/doc/html/tep116.html, 2006-06-27.

[13] O. Goldschmidt and D. S. Hochbaum, "Polynomial algorithm for the K-cut problem," in *IEEE 29th Annual Symposium on Foundations of Computer Science*, 1988, pp. 444–451.

[14] Y. Kamidoi, S. Wakabayashi, and N. Yoshida, "Faster algorithms for finding a minimum K-way cut in a weighted graph," in *1997 IEEE International Symposium on Circuits and Systems*, 1997.

[15] L. Zhao, H. Nagamochi, and T. Ibaraki, "Approximating the minimum K-way cut in a graph via minimum 3-way cuts," in *ISAAC '99: Proceedings of the 10th International Symposium on Algorithms and Computation*, 1999, pp. 373–382.

[16] J. W. Hui and D. Culler, "The dynamic behavior of a data dissemination protocol for network programming at scale," in *Proceedings of the 2nd international conference on Embedded networked sensor systems*, 2004, pp. 81–94.

[17] P. Levis, N. Lee, M. Welsh, and D. Culler, "Tossim: Accurate and scalable simulation of entire tinyos applications," in *Proceedings of ACM SenSys'03*, 2003.

[18] M. Zuniga and B. Krishnamachari, "Analyzing the transitional region in low power wireless links," USC Tech Report 04-823.

[19] http://www.tinyos.net/tinyos-2.x/doc/html/tutorial/usc-topologies.html.

[20] A. Galland, S. Abiteboul, A. Marian, and P. Senellart, "Corroborating information from disagreeing views," in *WSDM*, 2010, pp. 131–140.

[21] J. Pasternack and D. Roth, "Knowing what to believe (when you already know something)," in *International Conference on Computational Linguistics (COLING)*, 2010.

[22] D. Wang, T. Abdelzaher, and L. Kaplan, "On truth discovery in social sensing: A maximum likelihood estimation approach," *UIUC Technical Report*, 2011.

[23] J. Han, "Mining heterogeneous information networks by exploring the power of links," in *Proceedings of the 20th international conference on Algorithmic learning theory, ser. ALT'09. Berlin, Heidelberg: Springer-Verlag.* [Online]. Available: http://portal.acm.org/citation.cfm?id=1813231.1813235 2009, pp. 3–3.

[24] J. Pasternack and D. Roth, "Knowing What to Believe (when you already know something)," in *Proc. the International Conference on Computational Linguistics (COLING)*, 2010.

[25] J. M. Kleinberg, "Authoritative sources in a hyperlinked environment," *Journal of the ACM*, vol. 46, no. 5, pp. 604–632, 1999.

[26] S. Brin and L. Page, "The anatomy of a large-scale hypertextual Web search engine," *Computer Networks and ISDN Systems*, vol. 30, no. 1-7, pp. 107–117, 1998.

[27] X. Yin, P. S. Yu, and J. Han, "Truth Discovery with Multiple Conflicting Information Providers on the Web," *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, no. 6, pp. 796–808, 2008. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4415269

[28] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining*, 2005.

[29] J. Huang, "Color-spatial image indexing and applications," 1998, phD thesis, Cornell University.

[30] What's Invasive Community Data Collection Project, "http://whatsinvasive.com."

[31] A. Thiagarajan, J. Biagioni, T. Gerlich, and J. Eriksson, "Cooperative transit tracking using smart-phones," in *SenSys'10*, 2010, pp. 85–98.

[32] H. Le, D. Wang, H. Ahmadi, M. Y. S. Uddin, Y. H. Ko, T. Abdelzaher, O. Fatemieh, J. Pasternack, D. Roth, J. Han, H. Wang, L. Kaplan, B. Szymanski, S. Adali, C. Aggarwal, and R. Ganti, "Apollo: A data distillation service for social sensing," University of Illinois Urbana-Champaign, Tech. Rep., 2012.

[33] D. Wang, H. Le, T. Abdelzaher, and L. Kaplan, "On truth discovery in social sensing: A maximum likelihood estimation approach," in *Proc of IPSN*, 2012.

[34] "Zeromq, the intelligent transport layer," http://www.zeromq.org.

[35] S. Kotsiantis and P. Pintelas, "Recent advances in clustering: A brief survey," *WSEAS Transactions on Information Science and Applications*, vol. 1, no. 1, pp. 73–81, 2004.

[36] A. McCallum, K. Nigam, and L. H. Ungar, "Efficient clustering of high-dimensional data sets with application to reference matching," in *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD '00. [Online]. Available: http://doi.acm.org/10.1145/347090.347123 2000, pp. 169–178.

[37] "Apache mahout: Scalable machine learning and data mining," http://mahout.apache.org/.

[38] R. Szewczyk, E. Osterweil, J. Polastre, M. Hamilton, A. Mainwaring, and D. Estrin, "Habitat monitoring with sensor networks," *Communications of the ACM*, vol. 47, no. 6, pp. 34–40, 2004.

[39] N. Xu, S. Rangwala, K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin, "A wireless sensor network for structural monitoring," in *Proceedings of the ACM Conference on Embedded Networked Sensor Systems*, 2004.

[40] G. Simon, M. Maroti, A. Ledeczi, G. Balogh, B. Kusy, A. Nadas, G. Pap, J. Sallai, and K. Frampton, "Sensor network-based countersniper system," in *Proceedings of the ACM Conference on Embedded Networked Sensor Systems*, 2004.

[41] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "Tinydb: an acquisitional query processing system for sensor networks," *ACM Transactions on Database Systems*, vol. 30, no. 1, pp. 122–173, 2005.

[42] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: A scalable and robust communication paradigm for sensor networks," in *Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking (MobiCOM'00)*, 2000.

[43] G. Zhou, J. Stankovic, and S. Son, "The crowded spectrum in wireless sensor networks," in *Proceedings of the Third Workshop on Embedded Networked Sensors (EmNets 2006)*, 2006.

[44] P. Bahl, R. Chandra, and J. Dunagan, "SSCH: Slotted seeded channel hopping for capacity improvement in IEEE 802.11 ad-hoc wireless networks," in *Proceedings of ACM MobiCom'04*, 2004.

[45] A. Tzamaloukas and J. Garcia-Luna-Aceves, "Channel-hopping multiple access," in *Proceedings of IEEE ICC'00*, 2000.

[46] P. Kyasanur and N. H. Vaidya, "Routing and interface assignment in multi-channel multi-interface wireless networks," in *Proceedings of IEEE WCNC'05*, 2005.

[47] X. Chen, P. Han, Q.-S. He, S. liang Tu, and Z.-L. Chen, "A multi-channel MAC protocol for wireless sensor networks," in *Proceedings of The Sixth IEEE International Conference on Computer and Information Technology (CIT'06)*, 2006.

[48] http://www.xbow.com/Products/productdetails.aspx?sid=164.

[49] "Technical overview of time synchronized mesh protocol (tsmp)," TSMP White Paper, http://www.dustnetworks.com/, 2006.

[50] P. Kinney and D. Sexton, "Isa100.11a release 1 - an update on the process automation applications wireless standard," http://www.isa.org/isasp100/, 2008.

[51] J. A. Patel, H. Luo, and I. Gupta, "A cross-layer architecture to exploit multichannel diversity with a single transceiver," in *INFOCOM Minisymp. 2007*, 2007.

[52] J. Burke et al., "Participatory sensing," in *Workshop on World-Sensor-Web (WSW): Mobile Device Centric Sensor Networks and Applications*, 2006, pp. 117–134.

[53] T. Abdelzaher et al., "Mobiscopes for human spaces," *IEEE Pervasive Computing*, vol. 6, no. 2, pp. 20–29, 2007.

[54] J.-H. Huang, S. Amjad, and S. Mishra, "CenWits: a sensor-based loosely coupled search and rescue system using witnesses," in *SenSys'05*, 2005, pp. 180–191.

[55] B. Hull et al., "CarTel: a distributed mobile sensor computing system," in *SenSys'06*, 2006, pp. 125–138.

[56] S. B. Eisenman et al., "The bikenet mobile sensing system for cyclist experience mapping," in *SenSys'07*, November 2007.

[57] Sense Networks, "Cab Sense," http://www.cabsense.com.

[58] R. K. Balan, N. X. Khoa, and L. Jiang, "Real-time trip information service for a large taxi fleet," in *9th International Conference on Mobile Systems, Appliations, and Services (MobiSys 2011)*, 2011.

[59] X. Yin, J. Han, and P. S. Yu, "Truth discovery with multiple conflicting information providers on the web," *IEEE Trans. on Knowl. and Data Eng.*, vol. 20, pp. 796–808, June 2008. [Online]. Available: http://portal.acm.org/citation.cfm?id=1399100.1399392

[60] X. L. Dong, L. Berti-Equille, and D. Srivastava, "Integrating conflicting data: the role of source dependence," *Proc. VLDB Endow.*, vol. 2, pp. 550–561, August 2009. [Online]. Available: http://portal.acm.org/citation.cfm?id=1687627.1687690

[61] X. Dong, L. Berti-Equille, and D. Srivastava, "Truth discovery and copying detection in a dynamic world," *VLDB*, vol. 2, no. 1, pp. 562–573, 2009. [Online]. Available: http://portal.acm.org/citation.cfm?id=1687627.1687691

[62] L. Berti-Equille, A. D. Sarma, X. Dong, A. Marian, and D. Srivastava, "Sailing the information ocean with awareness of currents: Discovery and application of source dependence," in *CIDR'09*, 2009.

[63] X. Dong, L. Berti-Equille, Y. Hu, and D. Srivastava, "Global detection of complex copying relationships between sources," *PVLDB*, vol. 3, no. 1, pp. 1358–1369, 2010.

[64] L. Blanco, V. Crescenzi, P. Merialdo, and P. Papotti, "Probabilistic models to reconcile complex data from inaccurate data sources," in *CAiSE*, 2010, pp. 83–97.

[65] A. M. Truskinger, H. Yang, J. Wimmer, J. Zhang, I. Williamson, and P. Roe, "Large scale participatory acoustic sensor data analysis : tools and reputation models to enhance effectiveness," in *7th IEEE International Conference on e-Science*, B. Werner, Ed., December 2011.

[66] J. Pasternack and D. Roth, "Generalized fact-finding (poster paper)," in *World Wide Web Conference (WWW'11)*, 2011.

[67] M. Gupta, Y. Sun, and J. Han, "Trust analysis with clustering (poster paper)," in *20th World Wide Web Conference (WWW'11)*, 2011.

[68] R. Balakrishnan, "Source rank: Relevance and trust assessment for deep web sources based on inter-source agreement," in *20th World Wide Web Conference (WWW'11)*, 2011.

[69] X. Yin and W. Tan, "Semi-supervised truth discovery," in *WWW*, 2011.

[70] Y. Sun, Y. Yu, and J. Han, "Ranking-based clustering of heterogeneous information networks with star network schema," in *15th SIGKDD international conference on Knowledge discovery and data mining (KDD'09)*. [Online]. Available: http://doi.acm.org/10.1145/1557019.1557107 2009, pp. 797–806.

[71] D. Wang, H. Le, T. Abdelzaher, and L. Kaplan, "On truth discovery in social sensing: A maximum likelihood estimation approach," in *Submitted to IPSN 12*, 2012.

[72] D. Wang, T. Abdelzaher, L. Kaplan, and C. C. Aggarwal, "On quantifying the accuracy of maximum likelihood participant reliability estimation in social sensing," in *8th International Workshop on Data Management for Sensor Networks (DMSN 2011)*, 2011.

[73] M. Srivastava, T. Abdelzaher, and B. Szymanski, "Human-centric sensing," *Philosophical Transactions of the Royal Society, A*, vol. 370, no. 1958, pp. 176–197, 2012.

[74] B. Hull, V. Bychkovsky, Y. Zhang, K. Chen, M. Goraczko, A. Miu, E. Shih, H. Balakrishnan, and S. Madden, "CarTel: a distributed mobile sensor computing system," in *Proc of SenSys*, 2006.

[75] S. B. Eisenman, E. Miluzzo, N. D. Lane, R. A. Peterson, G.-S. Ahn, and A. T. Campbell, "The bikenet mobile sensing system for cyclist experience mapping," in *Prof of Sensys*, 2007.

[76] M. Y. Uddin, H. Wang, F. Saremi, G.-J. Qi, T. Abdelzaher, and T. Huang, "Photonet: a similarity-aware picture delivery service for situation awareness," in *Proc. of RTSS*, 2011.

[77] J.-H. Huang, S. Amjad, and S. Mishra, "Cenwits: a sensor-based loosely coupled search and rescue system using witnesses," in *Proc. of SenSys*, 2005, pp. 180–191.

[78] S. Reddy, A. Parker, J. Hyman, J. Burke, D. Estrin, and M. Hansen, "Image browsing, processing, and clustering for participatory sensing: Lessons from a dietsense prototype," in *Proc. of EmNets*, 2007.

[79] L. Berti-Equille, A. D. Sarma, X. Dong, A. Marian, and D. Srivastava, "Sailing the information ocean with awareness of currents: Discovery and application of source dependence," in *CIDR*, 2009.

[80] L. Blanco, V. Crescenzi, P. Merialdo, and P. Papotti, "Probabilistic models to reconcile complex data from inaccurate data sources," in *CAiSE*, 2010, pp. 83–97.

[81] A. Galland, S. Abiteboul, A. Marian, and P. Senellart, "Corroborating information from disagreeing views," in *Proceedings of the third ACM international conference on Web search and data mining*. [Online]. Available: http://portal.acm.org/citation.cfm?id=1718504 2010, pp. 131–140.

[82] X. L. Dong, L. Berti-Equille, and D. Srivastava, "Integrating conflicting data: The role of source dependence," *PVLDB*, vol. 2, no. 1, pp. 550–561, 2009.

[83] X. Dong, L. Berti-Equille, Y. Hu, and D. Srivastava, "Global detection of complex copying relationships between sources," *PVLDB*, vol. 3, no. 1, pp. 1358–1369, 2010.

[84] M. Gupta, Y. Sun, and J. Han, "Trust Analysis with Clustering," *cs.illinois.edu*, pp. 53–54, 2011. [Online]. Available: http://www.cs.illinois.edu/homes/gupta58/pp0280-gupta.pdf

[85] D. Wang, T. Abdelzaher, L. Kaplan, and C. C. Aggarwal, "On quantifying the accuracy of maximum likelihood participant reliability estimation in social sensing," in *8th International Workshop on Data Management for Sensor Networks (DMSN 2011)*, 2011.

[86] J. P. Callan, Z. Lu, and W. B. Croft, "Searching distributed collections with inference networks," in *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, ser. SIGIR '95, 1995, pp. 21–28.

[87] L. Gravano, H. Garcia-Molina, and A. Tomasic, "Gloss: text-source discovery over the internet," *ACM Transactions on Information Systems (TOIS)*, vol. 24, pp. 229–264, 1999.

[88] B. Yuwono and D. Lee, "Server ranking for distributed text retrieval systems on the internet," in *Proc of Database Systems for Advanced Applications*, 1997, pp. 41 – 49.

[89] F. Abbaci, J. Savoy, and M. Beigbeder, "A methodology for collection selection in heterogeneous contexts," in *Proc of Information Technology: Coding and Computing (ITCC)*, 2002.

[90] L. Si, R. Jin, J. Callan, and P. Ogilvie, "Language modeling framework for resource selection and results merging," in *Proc of Information and Knowledge Management (CIKM)*, 2002.

[91] D. Aksoy, "Information source selection for resource constrained environments," *SIGMOD Rec.*, vol. 34, no. 4, pp. 15–20, 2005.